

T 218328

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

IMPROVEMENTS TO IBMPIP

by

Dale A. Milton

March 1984

Thesis Advisor:

R. E. Ball

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Improvements to IBMPIP		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dale A. Milton		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE March 1984
		13. NUMBER OF PAGES 197
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Aircraft Survivability Interactive Graphics		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Two batch air defense programs named P001 and MICE reside on the Naval Postgraduate School IBM 3033 computer. The programs compute the probability of kill of an aircraft by defensive weapons. Data is fed to the programs by an interactive preprocessing program named IBMPIP. Improvements to IBMPIP are proposed to enhance existing capabilities or provide new capabilities. Areas addressed are: introduction of user friendliness, enhancement of graphics capabilities, improvement in the accuracy of results, and a suggested philosophy for evolutionary improvements.		

Approved for public release; distribution unlimited.

Improvements to IBMPIP

by

Dale A. Milton
Lieutenant, United States Navy
B.M.E., Auburn University, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1984

ABSTRACT

Two batch air defense programs named P001 and MICE reside on the Naval Postgraduate School IBM 3033 computer. The programs compute the probability of kill of an aircraft by defensive weapons. Data is fed to the programs by an interactive pre-processing program named IBMPIP. Improvements to IBMPIP are proposed to enhance existing capabilities or provide new capabilities. Areas addressed are:

- introduction of user friendliness
- enhancement of graphics capabilities
- improvement in the accuracy of results
- a suggested philosophy for evolutionary improvements

TABLE OF CONTENTS

I.	HISTORY AND INTRODUCTION	8
II.	USER FRIENDLINESS	13
III.	SUMMARY OF COMMONALITY BETWEEN PROGRAMS	18
IV.	SUMMARY OF CURRENT IBMPIP	21
V.	INTRODUCTION OF THE PROPOSED METHOD	33
	A. OVERVIEW	33
	B. BIGCAL LEVEL SUBROUTINES	36
	C. THIRD LAYER SUBROUTINES	57
VI.	NON-BIGCAL IBMPIP SUBROUTINES	65
	A. INTRODUCTION	65
	B. SUBROUTINES	65
	C. GRAPHIC CALLS	69
VII.	STATUS OF PROPOSED IBMPIP AND SUMMARY	72
APPENDIX A: USER'S MANUAL FOR CURRENT IBMPIP.		75
APPENDIX B: DERIVATION OF EQUATIONS USED IN TEXT		95
APPENDIX C: FLIGHT DYNAMICS AND COORDINATE TRANSFORMATIONS		102
APPENDIX D: CONSOLIDATED COMMON LISTING		112
APPENDIX E: EXECs USED TO DRIVE FORTRAN PROGRAMS		114
APPENDIX F: CLEANED UP JOHNS VERSION IBMPIP		115
APPENDIX G: PROPOSED IBMPIP LISTING		142



APPENDIX H: FLIGHTPAH GENERATOR WITHOUT THE	
GRAPHICS	181
LIST OF REFERENCES	196
INITIAL DISTRIBUTION LIST	197

LIST OF FIGURES

2.1	Continuous Flightpath Trace From Present IBMPIP	15
3.1	Elements Comprising Forward and Normal Acceleration	19
4.1	Invariants of the Current Trajectory	21
4.2	Spatial Components	22
4.3	Velocity Components	23
4.4	Angular Displacements	23
4.5	Computation of Spatial Components	24
4.6	Computation of Average Velocity (Speed)	25
4.7	Computation of Average Velocity Components	26
4.8	Computation of Average Acceleration Components	27
4.9	Components Required for Average Heading Vector	28
4.10	Transformation of Acceleration Components	29
5.1	Comparison of Schemes	34
5.2	Establishment of Move Vectors	38
5.3	Delta Vectors to the Temporary Point	41
5.4	Sense for Horizontal Turns	46
5.5	Sense for Pitch Above/Below the Horizon	47
5.6	Definition of Positive Rotations	49
5.7	Aircraft Rotational Velocities	52
5.8	Unrealistic Scenario	64
7.1	A Situation Not Addressed By Subroutine ACACC	72
C.1	The Map Coordinate System	102
C.2	The Aerodynamicist's Coordinate System	102

C.3	Components of the Equations of Motion	105
C.4	First Rotation	106
C.5	Second Rotation	107
C.6	Third Rotation	108
C.7	Finite Rotations Do Not Act As Vectors	109
C.8	Transformation of (ω)	110

I. HISTORY AND INTRODUCTION

There are two batch processor air defense simulation programs implemented at Naval Postgraduate School. These programs take a user-defined aircraft flight path, compare it to various defensive positions and return the probability of having killed the aircraft. The first program is the Air Force Armament Laboratory's Anti-Aircraft Artillery (AAA) Simulation. Its shorthand name is P001. The second program is the Air Force ASD surface-to-air missile (SAM) engagement simulation. Its shorthand name is MICE II. The programs satisfy MIL-STD-2069 and are used in industry as a design aid.

A drawback to these programs is the fact that the amount of input data submitted to the programs is large and the format requirements are rigid. The process of direct submission of data to the programs, is therefore time consuming and tedious. Humans are inefficient at performing time consuming, tedious tasks and errors are a result.

A first step improvement in the method of inputting data to P001 occurred in 1978. A pre-processing program named P001 Input Processor, or PIP for short, was created. Like the processors, PIP was a batch program. However, input requirements were reduced to just spatial coordinates of flightpath milestones, an initial speed, and control indicators for the type of output desired.

PIP had its shortcomings, too. Because of the batch implementation, the entire flightpath of milestones had to be submitted as a unit. The user learned of violations of aircraft structural/performance limitations and mission rules by experiencing failed runs. Corrections consisted of repeated tinkering with the milestones until a successful

CHAPTER 10

10.1.1. The first part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x}$ on the interval $(0, \infty)$.

10.1.2. The second part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^2}$ on the interval $(0, \infty)$.

10.1.3. The third part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^3}$ on the interval $(0, \infty)$.

10.1.4. The fourth part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^4}$ on the interval $(0, \infty)$.

10.1.5. The fifth part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^5}$ on the interval $(0, \infty)$.

10.1.6. The sixth part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^6}$ on the interval $(0, \infty)$.

10.1.7. The seventh part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^7}$ on the interval $(0, \infty)$.

10.1.8. The eighth part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^8}$ on the interval $(0, \infty)$.

10.1.9. The ninth part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^9}$ on the interval $(0, \infty)$.

10.1.10. The tenth part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^{10}}$ on the interval $(0, \infty)$.

10.1.11. The eleventh part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^{11}}$ on the interval $(0, \infty)$.

10.1.12. The twelfth part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^{12}}$ on the interval $(0, \infty)$.

10.1.13. The thirteenth part of the chapter is devoted to the study of the

properties of the function $f(x) = \frac{1}{x^{13}}$ on the interval $(0, \infty)$.

run emerged. A further drawback to PIP is the requirement for the user to plot the desired route, pick off numeric data and then get the numeric data into FORTRAN format.

The next generation improvement was introduced at Naval Postgraduate School in 1982 by Lt. Eric R. Johns. Recognizing the inherent limitations of a batch processor, Johns converted PIP into an interactive program suitable for use on time shared computer systems. In so doing, he realized the following benefits:

1. The user could be notified "immediately" of a milestone entry that violated predefined aircraft structural or performance limitations.
2. The user could be given the opportunity of making a correction before proceeding.
3. An interactive graphics capability could be introduced which eliminated the necessity of manually converting and coding data from a map into the computer.

As a secondary effort, transportability through modular programming was introduced to permit easy dissemination of the pre-processor to interested government agencies and industries. The version in use on the Naval Postgraduate School IBM 3033 computer is named IBMPIP.

While acknowledging the importance of IBMPIP as a great evolutionary step forward in the convenient use of P001 and MICE, it quickly became apparent that still more improvements could be realized, which would increase the convenience of the data generating aspect, improve the accuracy of the results from P001 and MICE and broaden the capabilities of the IBMPIP-P001/MICE combination in two areas: design decision making and tactics. The purpose of this thesis is to make some of those improvements.

Rather than rename IBMPIP, this thesis proposes an improved version of IBMPIP by installation of specific modules which are to be discussed. Physically, the introduction of the new modules required extensive changes to the existing modules.

Before developing the discussion for the proposed modules, it will be instructive to examine the use of IBMPIP-P001/MICE at Naval Postgraduate School. This acts as a first step toward developing an overall philosophy within which future modifications will fit in a logical sequence.

IBMPIP-P001/MICE is incorporated in a course titled Aircraft Combat Survivability (AE 3251) and is used as a teaching aid in developing an understanding of the factors affecting aircraft survivability in a hostile AAA and SAM environment. In a game environment, student teams "fly" a bombing mission against a target defended by seven AAA and one SAM locations. There are game restrictions on the location of some of the defensive positions. There are also game restrictions on the performance of the pseudo-aircraft. Each team is responsible for generating one defensive and one offensive data file. When each team is satisfied with their parameters, the data files are exchanged in such a manner that each team's flight path is compared to the other team's defense. Implicit in this method is the understanding that each team has no knowledge of what the incoming data files will look like. The comparisons, then, approach arbitrariness. P001 and MICE act as the judge, in that the team with the lowest probability of kill for their flightpath is named winner.

To ensure comparability of scores, the performance of the aircraft is fixed for the game, although IBMPIP presently has the capability of permitting performance and flight profile changes OF THE ONE aircraft described in it. This is an important point to note.

What is left as variables at the disposal of the teams then, are defensive positioning and the specific flightpath of the aircraft. This leads the students directly to the concept of tactics as a method of lowering probability of kill (Increasing chances of survival). This is good. What would be better would be for the student to reach the conclusion that reduction of vulnerability and susceptibility flaws in the aircraft will lead to a probability of survival exceeding that which can be accomplished by tactics alone. There are several modifications to IBMPIP that would accomplish this goal.

As stated earlier, P001 and MICE are used by industry to make design decisions. This fact leads to the demand that the data on the position of the aircraft be at least as accurate as the detect and track capability of the defensive devices programmed into P001 and MICE. Such is not presently the case. The original version of IBMPIP uses an assumed quadratic flightpath between the current milestone and the two previous milestones. The aircraft accelerations are computed at the center of these three milestones based on the velocity between the third and center milestones and the velocity between the center and current milestones. If the accelerations at the center milestone are acceptable, the current milestone is accepted. P001 and Mice assume a straight line between these milestones. Because the user is free to define milestones thousands of meters apart there can be a significant lateral error in the computed aircraft position at the mid-points between the milestones.

A last area to be discussed is the output from P001 and MICE. At present, it is difficult to obtain rapid correlation between the "aircraft" and where it is being subjected to attack. The output from both P001 and MICE are in the form of long tables of time-space-attitude information of the aircraft, probability of kills for each defensive

device and the cumulative probability of kill based on the sum of all of the defense. The user must then graph by hand the location of the "aircraft" at the time it is threatened or being damaged, to draw any conclusions on its susceptibility and to a lesser extent, its survivability.

By way of summary then, a statement is made as to what would constitute an ideal IBMPIP-P001/MICE implementation to satisfy all the above mentioned shortfalls. IBMPIP-P001/MICE should be:

- interactive to permit feed back of pertinent operating information to the user
- user friendly to diminish the requirement for specialized operating procedures and to avoid fatal errors
- approaching real time in ability to compute and present results
- graphics oriented to enhance the convenient generation of input data AND presentation of results
- accurate to the extent of the systems being modelled
- flexible through incorporation of a selectable range of different aircraft and defensive weapons
- transportable, as accomplished through modular programming techniques

In conclusion, two of the above points are addressed by this thesis and will be discussed. They are incorporation of user friendliness and a more representative flightpath. The most difficult task has been the convenient generation of an accurate, representative flight path.

II. USER FRIENDLINESS

A generally accepted meaning of "user friendliness" is to structure a program in such a fashion that the user cannot "do" anything to cause an unplanned halt in execution. The simplest version of the above concept is to prevent the user from entering data upon which the program is not structured to act. The following example is taken from the original IBMPIP.

```
50      WRITE (6,600)
        READ (5,*) IGUN
600     FORMAT('GUNS: 0=DISK FILE,1=TERMINAL,2=PRESET')
```

It is clear that the only correct responses are 0, 1, and 2. It is also clear that any integer number would satisfy the READ statement. To prevent unplanned events from taking place in the program, the following change is made:

```
50      WRITE (6,600)
        READ (5,*) IGUN
        IF(IGUN.NE.0.AND.IGUN.NE.1.AND.IGUN.NE.2)
          *GOTO 50
600     FORMAT(' 0=DISK FILE,1=TERMINAL,2=PRESET')
```

The user is now prevented from progressing beyond this point in the program until an acceptable input is made. The user is encouraged to try again by having the prompt message repeat on the screen.

If the definition of user friendliness is broadened to include "convenience" to the user, then there are many schemes which could profitably be incorporated. For example, in the current version of IBMPIP, once a map has been successfully created on the graphics screen, all further data is entered through a graphics buffer. The reason for this method is to allow the user to see a continuous trace (see figure 2.1) of the trajectory that will be submitted to P001/MICE. However, if a game parameter or an aircraft limitation is violated, the program suspends execution and queries the user for a selection of one of two options:

1. Accept the data point, which violates a parameter or,
2. Substitute a data point which does not.

This can turn into an iterative process between the user and the program.

It is a contention of this thesis that parameter violations which do not affect the overall outcome of the survivability of the "aircraft" should be corrected automatically by the computer. At most, for these kind of errors, the user should see an advisory statement that does not require a response. The following example is offered in support (to improve readability, some of the variables are renamed to simpler mnemonics).

One of the game parameters that must be satisfied is that during the enroute portion of the "flight", the "aircraft" may not descend below 60 meters altitude, as denoted by the variable HTMIN. Presented, then, is the original program flow for a violation of this parameter.

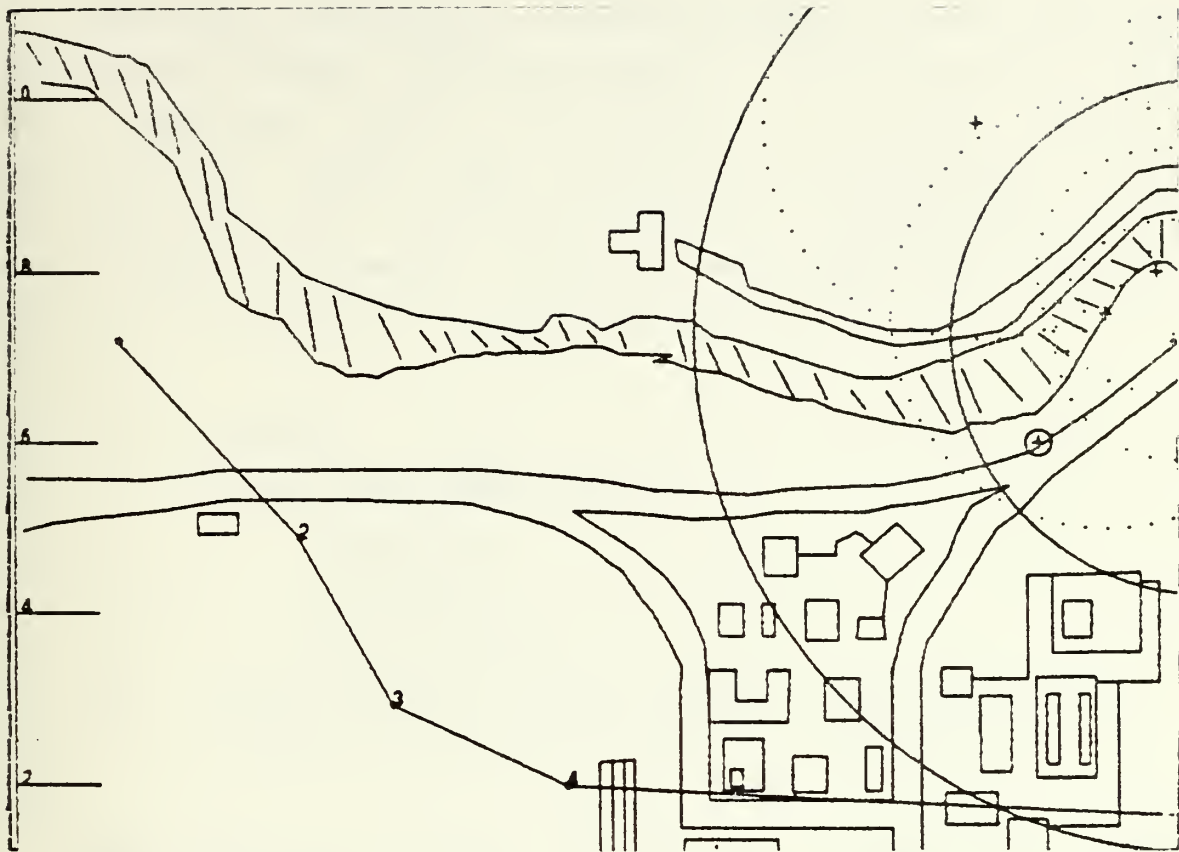


Figure 2.1 Continuous Flightpath Trace From Present IBMPIP.

```

1      ==>graphically input Z1
      ALT=Z1 (where Z1 is the altitude in meters
            -- call it 50 meters)
      IF(ALT.GE.HTMIN) proceed
            otherwise
            -ERROR=3
            GOTO (1,2,3,...),ERROR
3      WRITE (6,603)
603     FORMAT('ALTITUDE TOO LOW')
      WRITE (6,650)
      READ (5,*) ICOR
650     FORMAT('DO YOU WISH TO CORRECT ERROR,
            *1=YES,2=NO)
      IF(ICOR.EQ.1) GOTO 1

```


Positional parameters such as this are easily violated because the graphic setup constrains the "altimeter" to 200 meter increments and it is difficult to discern where 60 meters falls. If the user violated this parameter, it was obviously with the intent to go as low as possible without violating the parameter. It would seem reasonable, then, to modify the program in the following fashion:

```
ALT=Z1
IF(ALT.GE.HTMIN) proceed
    otherwise
        ALT=HTMIN
        WRITE(6,100)
100  FORMAT('USER VIOLATES MINIMUM ALT RESTRICTION:
        *COMPUTER CORRECTS; EXECUTION CONTINUES')
```

The following kind of situation is particularly frustrating, as there is no clue given to the user as to what is the problem. The statement is found in a passage that is reading in data from a pre-established disk file.

```
308  IF(MNUM.EQ.1.AND.LTR.EQ.83) STOP
```

MNUM..EQ.1 represents the first data point entered into the program. LTR is a control variable, and code 83 is that used to signify that "aircraft" milestone entry is complete. Unless the user happens to have access to the program listing, there is nothing to indicate that s/he is attempting to read in a flawed data set; yet selection of code 83 is permitted on the first data point entry when setting up the disk file.

The solution here is to simply route the program flow to a message informing the user of the difficulty.

```
308  IF(MNUM.EQ.1.AND.LIR.EQ.83) GOTO 100
100  FORMAT('INPUT DATA IS FATALLY FLAWED,
          *PROGRAM UNABLE TO CONTINUE, PROGRAM TERMINATES')
      STOP
```

In conclusion, user friendliness using concepts as stated above are incorporated into the improved version of IBMPIP. It is anticipated that as the scope of the users increases, more changes will become apparent.

III. SUMMARY OF COMMONALITY BETWEEN PROGRAMS

The current and proposed methods of generating a flight path share a common objective:

Accurately model the trajectory of a flight of a fixed-wing aircraft. Make the model realistic by subjecting the trajectory to the performance limitations of the real aircraft.

The aircraft performance limitations for both programs are:

1. acceleration along the longitudinal axis of the aircraft
2. acceleration normal to the longitudinal axis ("g" loading)

The forward acceleration is composed of components from:

1. engine thrust
2. aerodynamic drag
3. gravity

The normal acceleration is composed of components from:

1. lift
2. drag
3. gravity

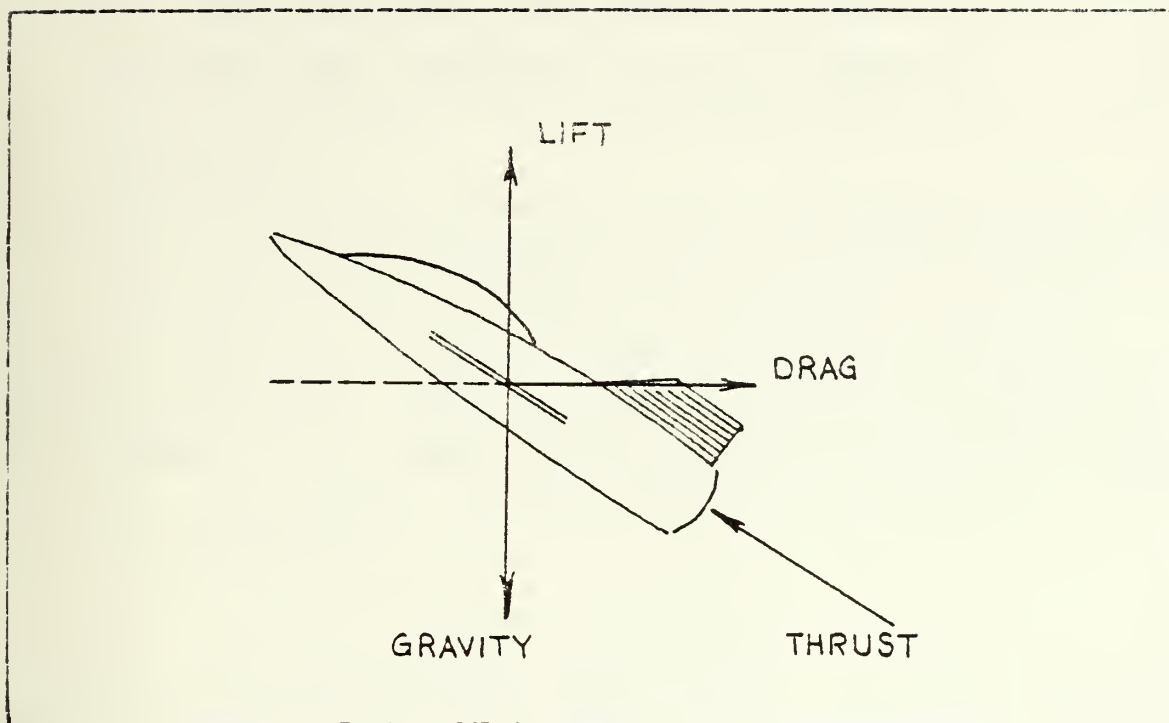


Figure 3.1 Elements Comprising Forward and Normal Acceleration.

Figure 3.1 illustrates the components making up forward and normal accelerations. There are ten parameters of the trajectory which must be determined for subsequent use by P001 and MICE. They are:

1. spatial components (x, y, z)
2. velocity components (u, v, w)
3. attitudinal components (heading angle (ψ) , pitch angle (θ) , and roll angle (ϕ))
4. elapsed time

The overall method by which the components are generated in the two programs are the same:

1. The ten components representing translation/rotation from one input milestone (x,y,z and velocity) to the next are computed in the earth-fixed (inertial) coordinate system.
2. The accelerations resulting from the displacement in the earth-fixed coordinate systems are transformed into the aircraft-fixed (rotational) coordinate system.
3. The thrust and g-load on the aircraft are computed.
4. If the thrust and g-loading fall within predefined limits, the 10 components are accepted, written to file and the program re-initialized to accept the user's next milestone.

One difference between the current and the proposed program occurs in the treatment of the conditions of over-thrust and over-g-load. The current program halts the execution and requests a user fix to the milestone. The proposed program adjusts the parameters internally until thrust and g-limitations are met.

IV. SUMMARY OF CURRENT IBMPIP

The current method is described below. Three milestones are used to compute average values of heading, climb and velocity components of the middle milestone. The flight path is always expressed as a straight line between two points. With this system, the invariables are the spatial coordinates of the milestones and the "aircraft" speed at

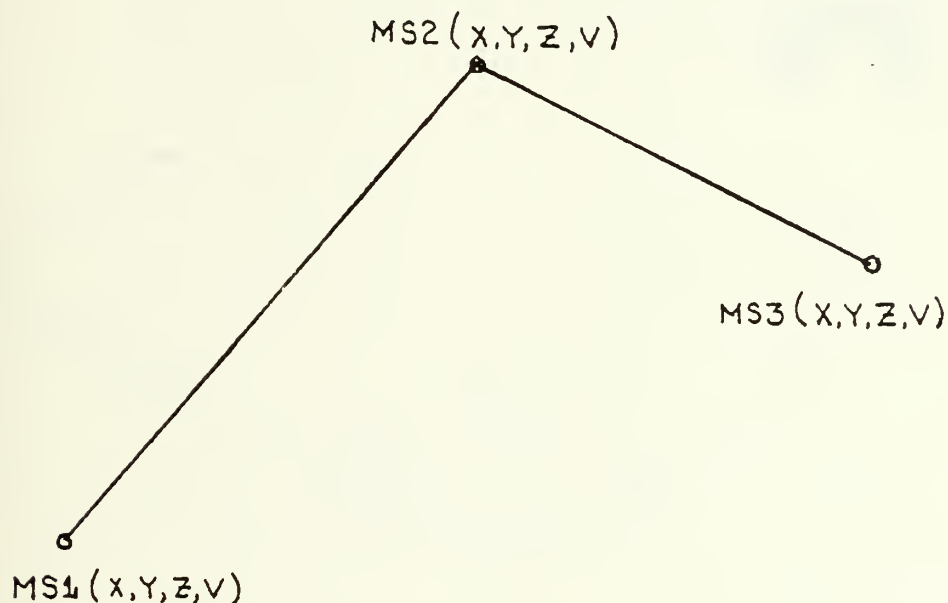


Figure 4.1 Invariants of the Current Trajectory.

the waypoints. As the trace in figure 4.1 is generated, two tasks are accomplished:

1. The user gets a "feel" for where the aircraft is going on the map.
2. Data is computed on the ten aircraft parameters, which at the completion of a run of IBMPIP, will be submitted to the batch programs P001 and/or MICE.

The data required by P001 and MICE are presented pictorally in the following figures.

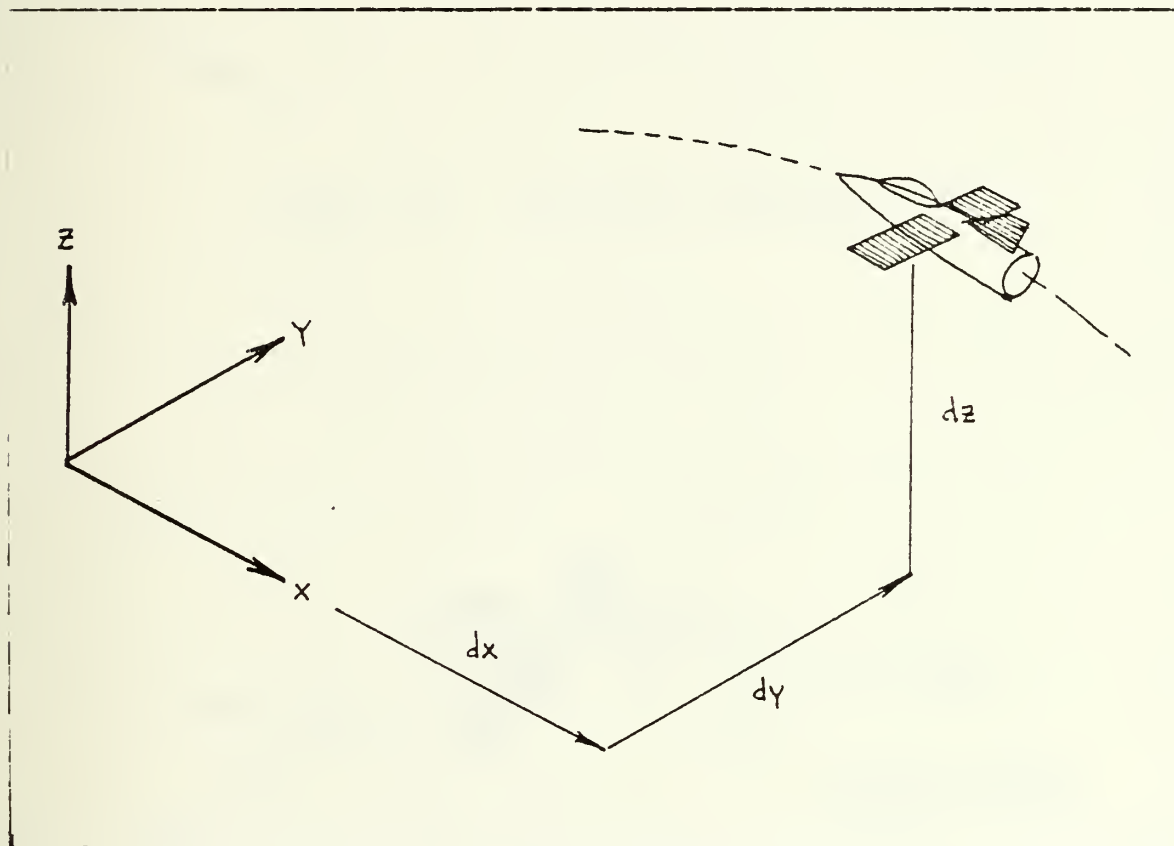


Figure 4.2 Spatial Components.

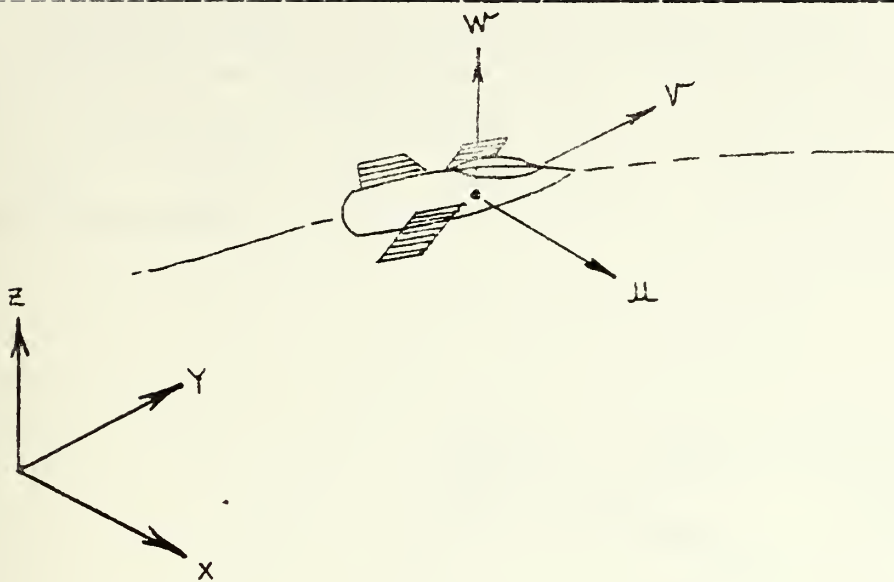


Figure 4.3 Velocity Components.

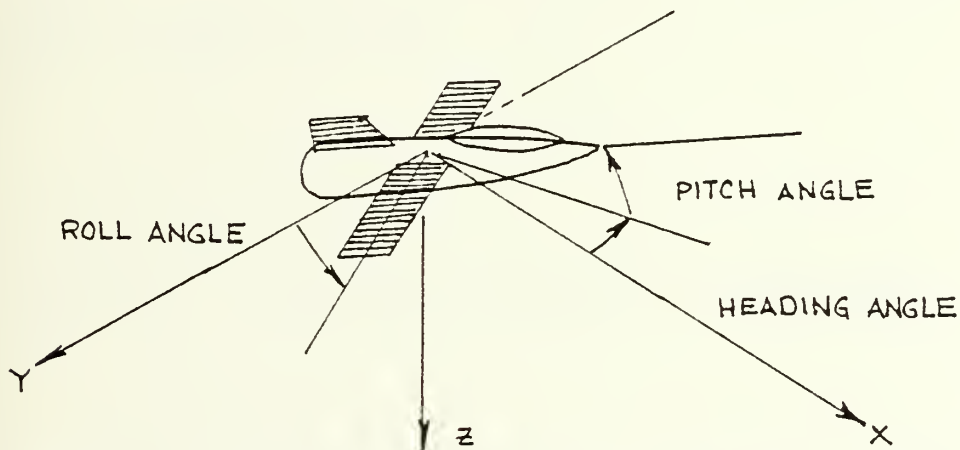


Figure 4.4 Angular Displacements.

The method used to generate the above data will now be presented in the following figures. To make viewing convenient, all diagrams will occupy the X-Y plane with the understanding that the discussion applies to the three-dimensional case.

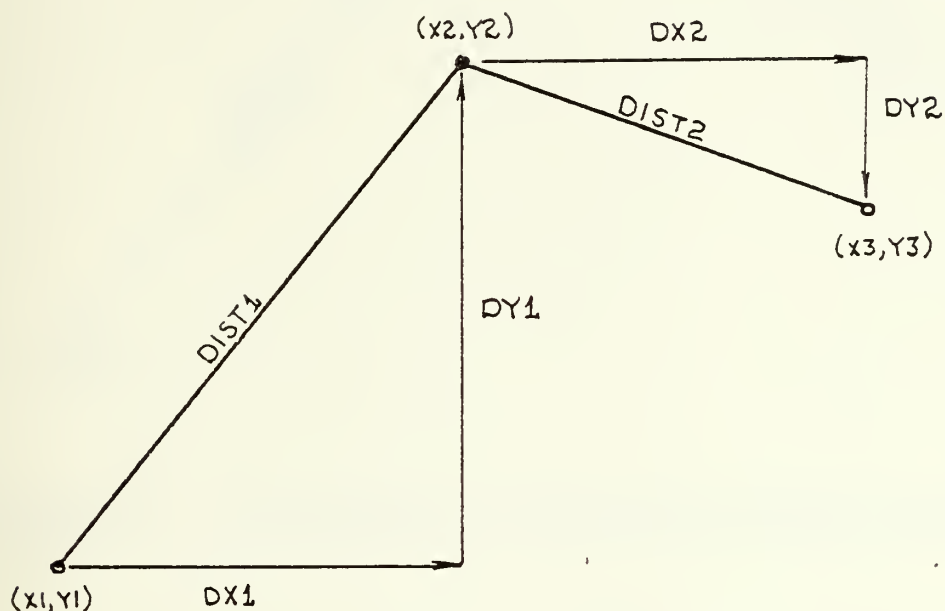


Figure 4.5 Computation of Spatial Components.

$$DX1 = (X2 - X1) \quad DX2 = (X3 - X2)$$

$$DY1 = (Y2 - Y1) \quad DY2 = (Y3 - Y2)$$

$$DZ1 = (Z2 - Z1) \quad DZ2 = (Z3 - Z2)$$

$$DIST1 = \text{SQRT}(DX1**2 + DY1**2 + DZ1**2)$$

$$DIST2 = \text{SQRT}(DX2**2 + DY2**2 + DZ2**2)$$

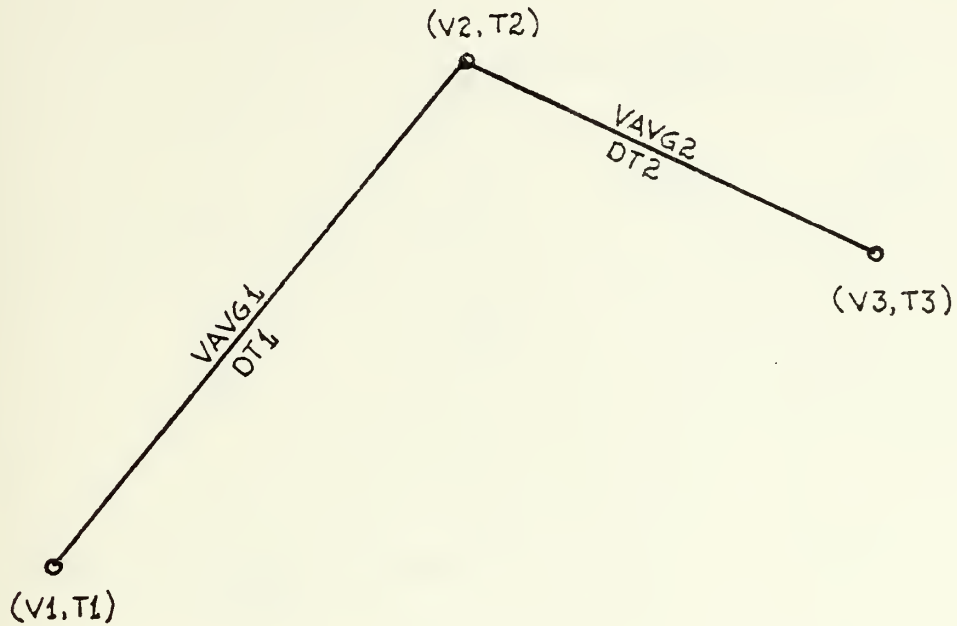


Figure 4.6 Computation of Average Velocity (Speed).

$$VAVG1 = (V2 - V1) / 2$$

$$VAVG2 = (V3 - V2) / 2$$

The elapsed time for the leg and total to the milestone are computed as follows:

$$DT1 = DIST1 / VAVG1$$

$$DT2 = DIST2 / VAVG2$$

$$T2 = T1 + DT1$$

$$T3 = T2 + DT2$$

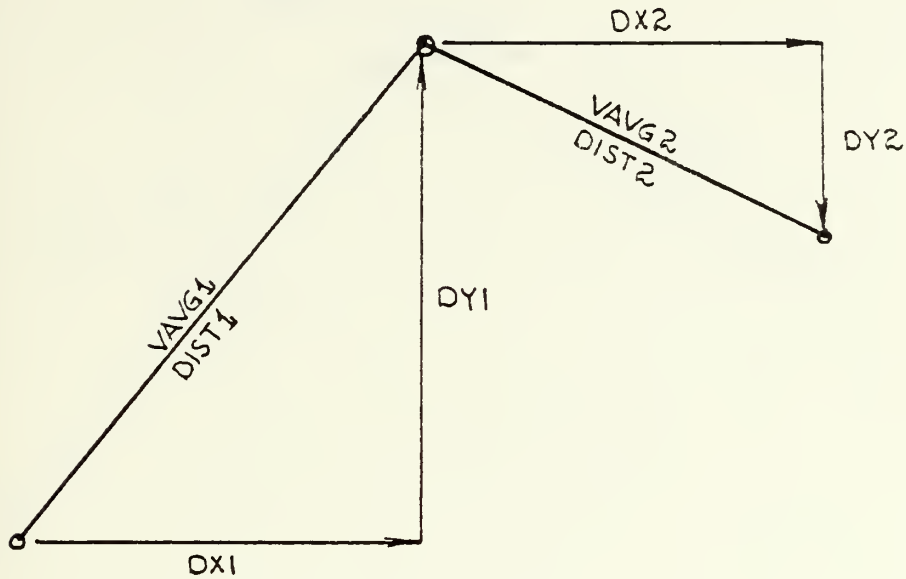


Figure 4.7 Computation of Average Velocity Components.

$$VX1 = VAVG1 * (DX1 / DIST1)$$

$$VY1 = VAVG1 * (DY1 / DIST1)$$

$$VZ1 = VAVG1 * (DZ1 / DIST1)$$

$$VX2 = VAVG2 * (DX2 / DIST2)$$

$$VY2 = VAVG2 * (DY2 / DIST2)$$

$$VZ2 = VAVG2 * (DZ2 / DIST2)$$

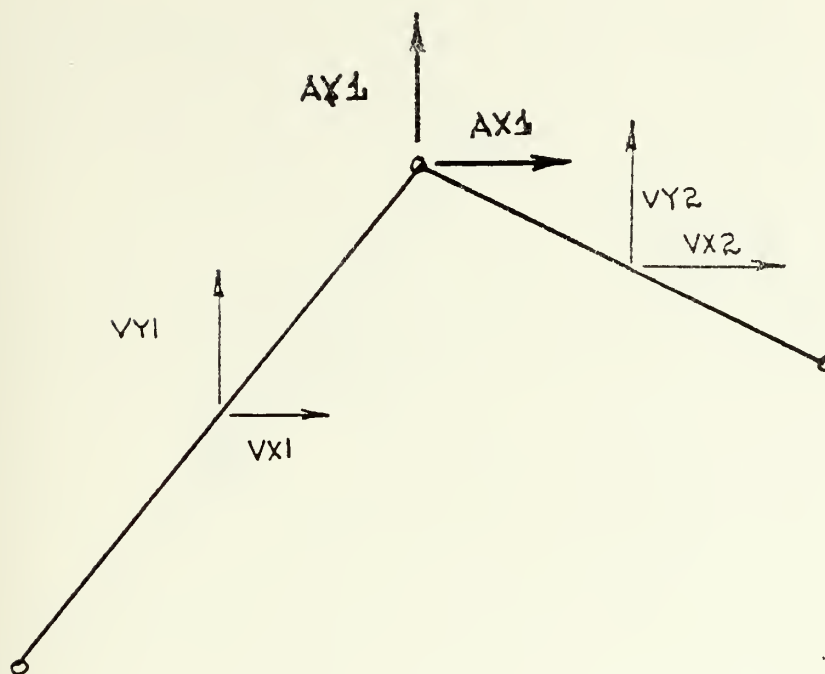


Figure 4.8 Computation of Average Acceleration Components.

$$\text{DELTAT} = (T3 - T1) / 2$$

$$A_{X1} = (V_{X2} - V_{X1}) / \text{DELTAT}$$

$$A_{Y1} = (V_{Y2} - V_{Y1}) / \text{DELTAT}$$

$$A_{Z1} = (V_{Z2} - V_{Z1}) / \text{DELTAT}$$

When at milestone two, the aircraft is presumed to have an attitude which is an average of the attitude at milestone one (MS1) and the attitude at milestone two (MS2). Unit vectors are generated at MS2 using the following weighting scheme.

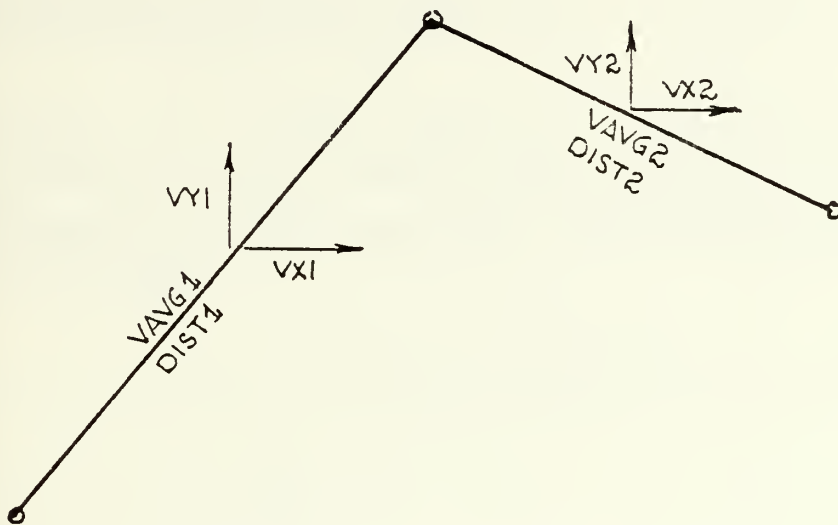


Figure 4.9 Components Required for Average Heading Vector.

$$HDGAVG = DIST2 / (DIST1 + DIST2)$$

$$TDX = ((VX2/VAVG2) - (VX1/VAVG1)) * HDGAVG + VX1/VAVG1$$

$$TDY = ((VY2/VAVG2) - (VY1/VAVG1)) * HDGAVG + VY1/VAVG1$$

$$TDZ = ((VZ2/VAVG2) - (VZ1/VAVG1)) * HDGAVG + VZ1/VAVG1$$

The vectors are converted to direction cosines:

$$UNIT = \sqrt{TDX^2 + TDY^2 + TDZ^2}$$

$$TDX = TDX / UNIT$$

$$TDY = TDY / UNIT$$

$$TDZ = TDZ / UNIT$$

Rotational angles can now be defined:

$$\text{HDG} = \text{ATAN}(\text{TDY}/\text{TDX})$$

$$\text{CA} = \text{ATAN}(\text{TDZ}/\text{SQRT}(\text{TDX}^2 + \text{TDY}^2))$$

Computation of the roll angle is temporarily deferred. The accelerations in the earth-fixed coordinate system must be transformed to the rotational coordinate system (see figure 4.10) below.

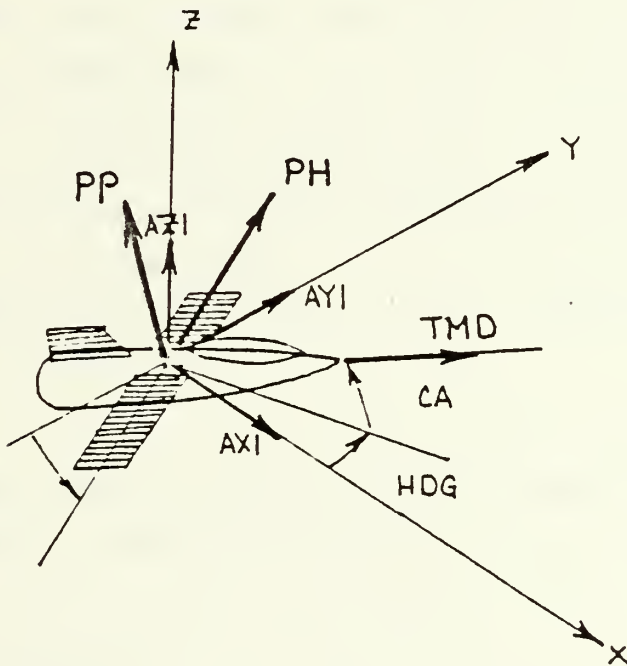


Figure 4.10 Transformation of Acceleration Components.

The coordinate transformations are given without development [Ref. 1]

$$TMD = \begin{Bmatrix} \cos(HDG) * \cos(CA) \\ \sin(HDG) * \cos(CA) \\ \sin(CA) \end{Bmatrix}^T \begin{Bmatrix} AX1 \\ AY1 \\ AZ1 + GEE \end{Bmatrix}$$

$$PH = \begin{Bmatrix} \sin(HDG) \\ \cos(HDG) \\ 0 \end{Bmatrix}^T \begin{Bmatrix} AX1 \\ AY1 \\ AZ1 + GEE \end{Bmatrix}$$

$$PP = \begin{Bmatrix} \cos(HDG) * \sin(CA) \\ \sin(HDG) * \sin(CA) \\ \cos(CA) \end{Bmatrix}^T \begin{Bmatrix} AX1 \\ AY1 \\ AZ1 + GEE \end{Bmatrix}$$

TMD= acceleration along the aircraft's longitudinal axis

PH= acceleration in the direction of the aircraft's right wing

PP= acceleration in the direction of the top of the aircraft

GEE=acceleration due to the earth's gravity (9.82 m/s²)

The roll angle of the aircraft with respect to the earth-fixed coordinate system is written:

$$RA = -\text{ATAN}(PH/PP)$$

The total aircraft lift is the vector sum of the accelerations normal to forward translation and is normal to the roll angle:

$$ACLIFT = \text{SQRT}(PH**2 + PP**2)$$

Both the forward translation acceleration and the lift or normal acceleration are non-dimensionalized by the gravity term:

$$TMD1 = TMD / GEE$$

$$ACLIFT1 = ACLIFT / GEE$$

The above is based upon the concept of force divided by weight:

$$THRUST / WEIGHT = m(TMD) / mg = TMD / GEE = TMD1$$

Now aerodynamic drag on the airframe must be accounted for. Drag is computed using conventional aerodynamic equations. Computation of air density as a function of altitude:

$$RHO = .256 * EXP(-.103 * Z2 / 1000)$$

Z2 = absolute altitude of milestone 2 in meters

This equation is in the non-standard dimensions (lbf-sec²/ft²-meter²) and is therefore unacceptable. It will be supplanted in the following chapter.

Computation of the coefficient of lift

$$CL = 2 * ACLIFT / (RHO * V2^{**2} / WL)$$

WL = wing loading in lbf/ft²

Computation of drag (non-dimensional):

$$DW = ((CDO + CDK * CL^{**2} / CL) * ACLIFT$$

CDO= coefficient of drag at zero angle of attack

$CDK * CL^{**2}$ = drag due to lift

Net longitudinal acceleration factor:

$$TW = TMD + DW$$

Speed brakes are accounted for by adding a surface area factor to the drag equation:

$$DW(s.b.) = ((0.05 + CDO + CDK * CL^{**2}) / CL) * ACLIFT$$

0.05 = 5% of the wing surface area

The limiting parameters are then:

1. TW
2. ACLIFT

7. INTRODUCTION OF THE PROPOSED METHOD

A. OVERVIEW

The proposed trajectory modules were written to satisfy the following goals:

1. Retain user friendliness features. This is defined in the following fashion: once a user enters a waypoint (we are here renaming milestones) through the graphics, the computer should fit a flight path to that waypoint without any further action on the part of the user. Interpret this to mean the user cannot enter a fatal waypoint.
2. The trajectory should closely match an actual aircraft's in spatial coordinates, linear velocities, and attitudinal rotations from the inertial coordinate system.

Observe figure 5.1 for a comparison between the two systems.

Under the proposed system the following gross events take place:

1. The aircraft is defined at a beginning waypoint which is named "old".
2. The user enters a next-waypoint, defined as "new".
3. Linear acceleration components are computed from "old" to "new".
4. Using the accelerations and an invariant increment of time, the aircraft is moved from "old" to a new point defined as "temp".

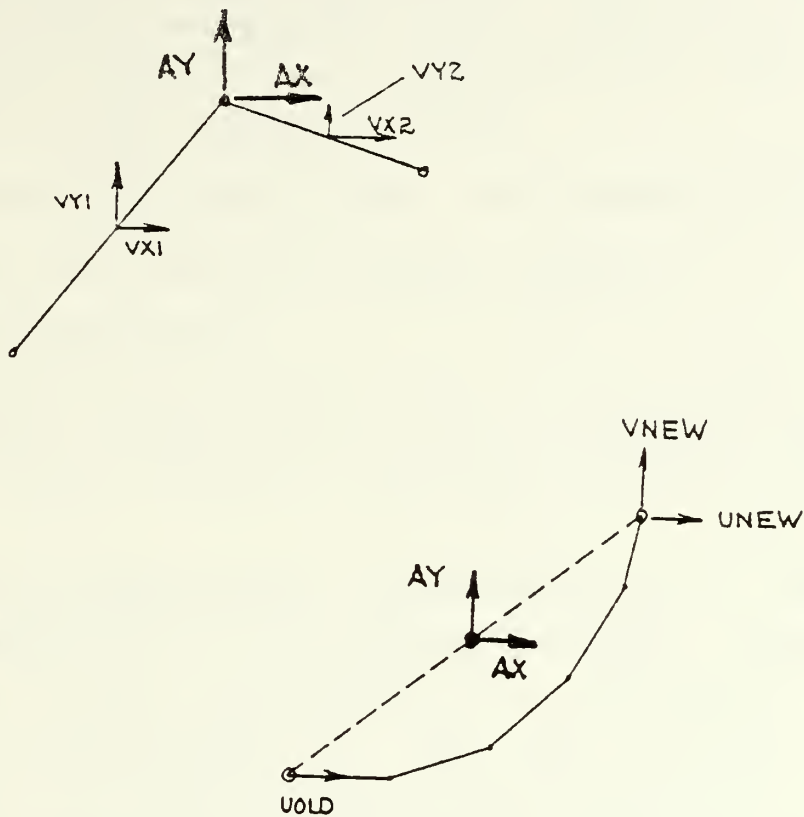


Figure 5.1 Comparison of Schemes.

5. Knowing the displacement vectors that got us to "temp", velocity components at "temp" are computed.
6. Knowing the displacement vectors that got us to "temp", angular displacements are computed
7. Knowing the time increment, rotational velocities are computed.
8. The linear AND rotational accelerations at "temp" are transformed into aircraft-fixed coordinates.
9. The transformed accelerations are fed into the left hand side (LHS) of the Aircraft Equations of Motion.

10. If thrust and g-load limitaitons are not exceeded, "temp" becomes an accepted point, the parameters are written to file, the map is updated by one time increment and "temp" becomes new "old".
11. The process is repeated until "old" becomes "new".
12. The user is queried for the next waypoint.

The new features are:

1. The computer flight path is now a discretized version of an actual flightpath and hence is more realistic model.
2. Rotational accelerations are accounted for. They are not accounted for in the present program. Thus, the present aircraft performance is artificially better than the performance of the real aircraft being modeled. Survivability data is too optimistic.

Imposed on the above events is an iteration that does two things:

1. If accelerations resulting from the user input lead to "aircraft" thrust and g-loadings less than the maximum defined, the accelerations are increased until thrust or g-loading hits maximum (whichever occurs first).
2. If thrust or g-loading are too great, either the user input velocity is adjusted downward or the "new" waypoint is moved in a straight line away from the "old" waypoint, until thrust AND g-load fall within limitations.

Number one was included under the philosophy that a pilot flying an aircraft which is being shot at will not be flying at less than the maximum available. A drawback to the current system is the user has no way of knowing (except by

exceeding limitations) whether the aircraft is at maximum performance or not. Number two was included in keeping with the desire to model the real aircraft as closely as possible. The hierarchy is based upon informal interviews with attack pilots attending NPS.

The rest of the Chapter contains a detailed examination of the subroutines and an explanation of the variables related to flightpath generation. All equations in the examinations are either dimensionless or in SI units. The subsequent Chapter reviews the remaining subroutines of IBMPIP. The final Chapter summarizes the current status of the proposed IBMPIP.

B. BIGCAL LEVEL SUBROUTINES

1. BIGCAL

- a. expanded name: Big Call
- b. function: Big Call acts as a driver for the rest of the subroutines involved with flight trajectory generation.
- c. equations/calls:

```
10      CALL TRNFRM
        CALL NEWOLD
        CALL MVELOS
        CALL MVDOTS
        CALL DELVEC
        CALL TMPPNT
        CALL TMPVEL
        CALL INRTRM
        CALL VECTOR
        CALL ANGLES
        CALL ROTRAT
        CALL ACTRM1
        CALL ACTRM2
```



```
CALL ACTRM3
CALL ACRVEL
CALL ACACC
CALL ACLMTS
CALL XCHNGE
CALL WAYPNT
CALL FILE
CALL GRAFIX
```

d. comments: The two advantages in this approach are the ability to treat all the flight trajectory subroutines as a single package and having this package accessed by a single call from the main program of IEMPIP.

2. TRNFRM

- a. expanded name: Transform
- b. function: Transform takes the user defined values which are acquired from the graphics buffer and loads them into variables peculiar to the aircraft trajectory package.
- c. equations/calls:

```
XNEW= X1
YNEW= Y1
ZNEW= Z1
VELCN=V1
```

d. comments: The above is done to permit "in-house" manipulation of the waypoint values without affecting the non-BIGCAL subroutines found elsewhere in IBMPIP. Note that in effect, this is the definition of waypoint "new" for the trajectory package.

3. NEWCLD

- a. expanded name: from Old to New
- b. function: The generation of shorter variable names for the convenience of the programmer.
- c. equations/calls:

```
AG1=XNEW-XOLD  
AH1=YNEW-YOLD  
AI1=ZNEW-ZOLD  
AJ1=SQRT(AG1**2+AH1**2+AI1**2)  
AJ2=SQRT(AG1**2+AH1**2)
```

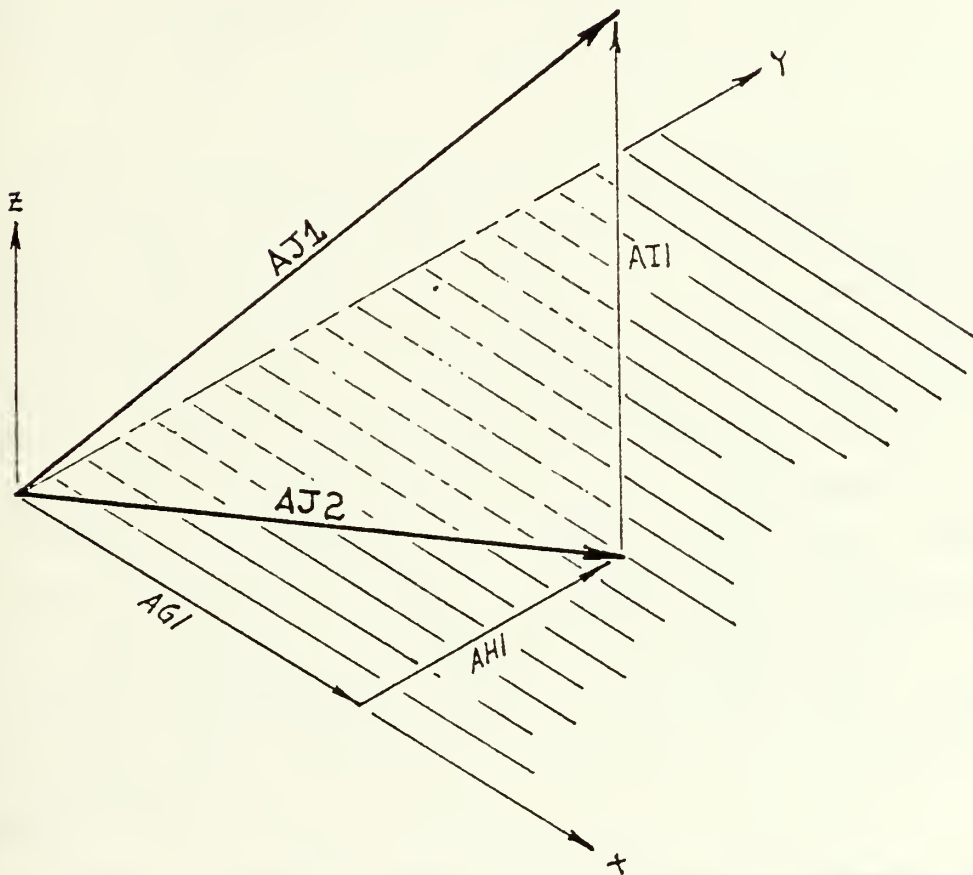


Figure 5.2 Establishment of Move Vectors.

d. comments: See figure 5.2. Note that AJ2 is the shadow of AJ1 on the x-y plane.

4. MVELOS

- a. expanded name: Map Coordinate Velocity Components
- b. function: Generates orthogonal velocity components for the user supplied speed at waypoint "new".
- c. equations/calls:

```
IF(N.GT.0) GOTO 10
      UNEW=VELON*(AG1/AJ1)
      VNEW=VELON*(AH1/AJ1)
      WNEW=VELON*(AI1/AJ1)
10    CONTINUE
      N=1
```

d. comments: The components are based upon the direction cosines from "old" to "new". As the "aircraft" moves from "old" to "new" in the subsequent programs, the direction cosines can change, sometimes drastically. Those changes are taking place along a temporary path which is "fuzzy" at first. It is fatal to the program for this "fuzziness" to be allowed to influence the endpoint velocity components (located at "new"). Therefore, the computation of the velocity components is restricted to one iteration between any given "old" and "new". There are some well defined exceptions will be discussed downstream.

5. MVDCTS

- a. expanded name: Map Coordinate Velocity-Dot Components
- b. function: Establish the linear accelerations between waypoint "old" and "new".
- c. equations/calls:


```

ACC=(VELON**2-VELOO**2)/(2*AJ1)
UDOT=ACC*(AG1/AJ1)
VDOT=ACC*(AH1/AJ1)
WDOT=ACC*(AI1/AJ1)

```

- d. comments: Again the components are based upon the direction cosines. The derivation of the equation

$$ACC=(VEICN**2-VELOO**2)/(2*AJ1)$$

is given in appendix B.

6. DELVEC

- a. expanded name: Delta Vectors
- b. function: Compute the components of a ds vector in preparation of a move of the "aircraft" from "old" to a location "temp". "Temp" is a point along a curvilinear trace with the defined endpoints "old" and "new". Establishment of the points "temp" define the curvilinear trace.
- c. equations/calls:

```

DELX=UOLD*DELTEE+.5*UDOT*DELTEE**2
DELY=VOLD*DELTEE+.5*VDOT*DELTEE**2
DELZ=WOLD*DELTEE+.5*WDOT*DELTEE**2
IF (ABS(DELX).GT.ABS(AG1)) DELX=AG1
IF (ABS(DELY).GT.ABS(AH1)) DELY=AH1
IF (ABS(DELZ).GT.ABS(AI1)) DELZ=AI1

```

- d. comments: The equations

```

DELX=UOLD*DELTEE+.5*UDOT*DELTEE**2
DELY=VOLD*DELTEE+.5*VDOT*DELTEE**2
DELZ=WOLD*DELTEE+.5*WDOT*DELTEE**2

```


are conventional dynamic equations and are derived in appendix E. The term DELTEE is an invariant increment of time and is set for the program. The current value is one second and is set in BLOCK DATA. Thus for a time increment of one second, the

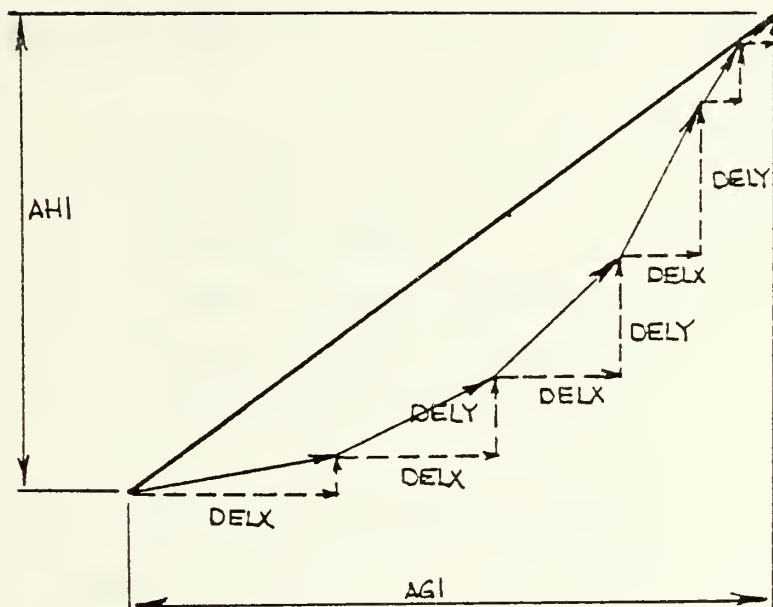


Figure 5.3 Delta Vectors to the Temporary Point.

aircraft is moved forward from "old" toward "new" by DEL amounts in the three orthogonal directions, utilizing the velocities and accelerations just computed. See figure 5.3. The logic statements

```
IF (ABS (DELX) .GT. ABS (AG1) ) DELX=AG1
IF (ABS (DELY) .GT. ABS (AH1) ) DELY=AH1
IF (ABS (DELZ) .GT. ABS (AI1) ) DELZ=AI1
```


deal with the situation where the "aircraft" is sufficiently close to "new" that the amount of translation covered in one second is greater than that needed to exactly arrive at "new" (prevents overshoot).

7. IMPENT

- a. expanded name: Temporary Point
- b. function: Define in map coordinates the location of the point "temp".
- c. equations/calls:

```
XTEMP=XOLD+DELX
YTEMP=YOLD+DELY
ZTEMP=ZOLD+DELZ
IF (DELX.EQ.AG1) XTEMP=XNEW
IF (DELY.EQ.AH1) YTEMP=YNEW
IF (DELZ.EQ.AI1) ZTEMP=ZNEW
```

- d. comments: The first three equations

```
XTEMP=XCID+DELX
YTEMP=YCID+DELY
ZTEMP=ZCID+DELZ
```

are self-explanatory. The group of logic statements

```
IF (DELX.EQ.AG1) XTEMP=XNEW
IF (DELY.EQ.AH1) YTEMP=YNEW
IF (DELZ.EQ.AI1) ZTEMP=ZNEW
```

are in keeping with the need for exact arrival at "new".

8. IMPVEL

- a. expanded name: Temporary Point Velocity Components

b. function: Generate velocity components for the point "temp".

c. equations/calls:

```
UTEMP=UOLD+UDOT*DELTEE
VTEMP=VOID+VDOT*DELTEE
WTEMP=WOLD+WDOT*DELTEE
IF (DELX.EQ.AG1) UTEMP=UNEW
IF (DELY.EQ.AH1) VTEMP=VNEW
IF (DELZ.EQ.AI1) WTEMP=WNEW
VELOT=SQRT (UTEMP**2+VTEMP**2+WTEMP**2)
```

d. comments: The below equations

```
UTEMP=UCID+UDOT*DELTEE
VTEMP=VCID+VDOT*DELTEE
WTEMP=WCID+WDOT*DELTEE
VELOT=SQRT (UTEMP**2+VTEMP**2+WTEMP**2)
```

are conventional dynamic equations and are derived in appendix E. The three logic statements

```
IF (DELX.EQ.AG1) UTEMP=UNEW
IF (DELY.EQ.AH1) VTEMP=VNEW
IF (DELZ.EQ.AI1) WTEMP=WNEW
```

again deal with the exact arrival at "new".

9. INRIRM

a. expanded name: Initial Rotational Transformation

b. function: Convert velocities and accelerations which are defined in terms of map coordinate (y points North, x points East, and z points "up") components to a coordinate system which sees conventional use by aircraft designers (y points South, x points East, and z points "down" toward the earth). See appendix C.

c. equations/calls:

```
U1= UTEMP
V1=-VTEMP
W1=-WTEMP
UD1= UDOT
VD1=-VDOT
WD1=-WDOT
```

d. comments: None.

10. ANGLES

- a. expanded name: Angle Change Increments
- b. function: Compute the angles resulting from the "aircraft's" move from "old" to "temp".
- c. equations/calls:

```
DS= SQRT (DELX**2+DELY**2+DELZ**2)
IF (DELX.NE.0.) GOTO 10
      IF (DELX.EQ.0..AND.DELY.LT.0.)
*      PSINEW=AA1
      IF (DELX.EQ.0..AND.DELY.GT.0.)
*      PSINEW=-AA1
      GOTO 20
10    CONTINUE
      PSINEW=-ATAN (DELY/DELX)
      IF (DELX.GT.0..AND.DELY.EQ.0.) PSINEW=0.
      IF (DELX.LT.0..AND.DELY.LT.0.)
*      PSINEW=AB1+PSINEW
      IF (DELX.LT.0..AND.DELY.EQ.0.) PSINEW=AB1
      IF (DELX.LT.0..AND.DELY.GT.0.)
*      PSINEW=PSINEW-AB1
20    CONTINUE
      THENEW=ASIN (DELZ/DS)
```



```

      TRNRAT=(P SINew-PSIOLD)/DELTEE
      IF (PSINew.GT.0.) PHINew=
*      ATAN (ABS (TRNRAT*VELOT/9.7958))
      IF (PSINew.LT.0.) PHINew=
*      -ATAN (ABS (TRNRAT*VELOT/9.7958))
      IF (PSINew.EQ.PSIOLD) PHINew=PHIOLD

```

d. comments: The aircraft has a particular attitude at the defined location "old". The movement to "temp" represents a "turn" toward "new". The angles are required for two reasons: the fact that the vulnerable area of the aircraft is a function of the aircraft's attitude with respect to damage propagators and the rotation of the aircraft in the turn generates additional accelerations which superpose on the linear ones just determined. These rotational accelerations must be accounted for when analysing the loads imposed on the aircraft. The variables

AA1=1.5708 radians (90 degrees)

AB1=3.1416 radians (180 degrees)

represent recurring quadrant angles. The equations

```

      IF (DELX.NE.0.) GOTO 10
      IF (DELX.EQ.0..AND.DELY.LT.0.)
*      PSINew=AA1
      IF (DELX.EQ.0..AND.DELY.GT.0.)
*      PSINew=-AA1
      GOTO 20
10      CONTINUE
      PSINew=-ATAN (DELY/DELX)
      IF (DELX.GT.0..AND.DELY.EQ.0.) PSINew=0.
      IF (DELX.LT.0..AND.DELY.LT.0.)
*      PSINew=AB1+PSINew

```



```

      IF (DELX.LT.0..AND.DELY.EQ.0.) PSINEW=AB1
      IF (DELX.LT.0..AND.DELY.GT.0.)
*      PSINEW=PSINEW-AB1

```

determine the turn angle PSINEW. The turn angle (or heading change) is referenced from the positive

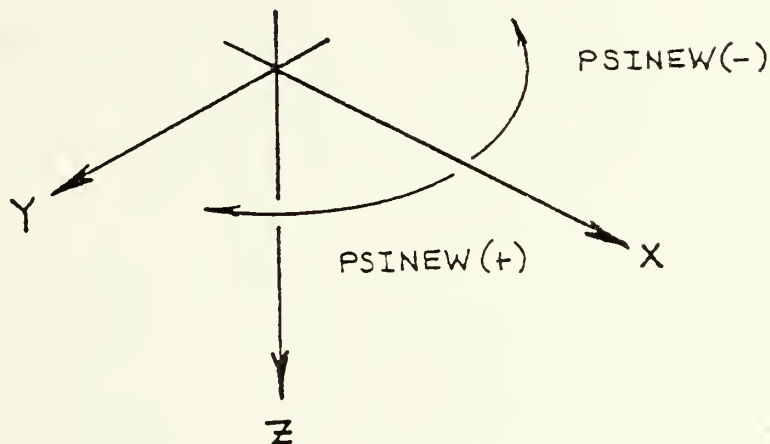


Figure 5.4 Sense for Horizontal Turns.

x-axis, is measured in radians, and may be either positive or negative. Positive is defined as a CLOCKWISE rotation. This sense is specified to insure compatibility with upcoming transformation matrices. See figure 5.4. The equation

$$THENEW=ASIN(DELZ/DS)$$

defines an angle THENEW (thetaneu) which is a measure of the "aircraft's" pitch above or below

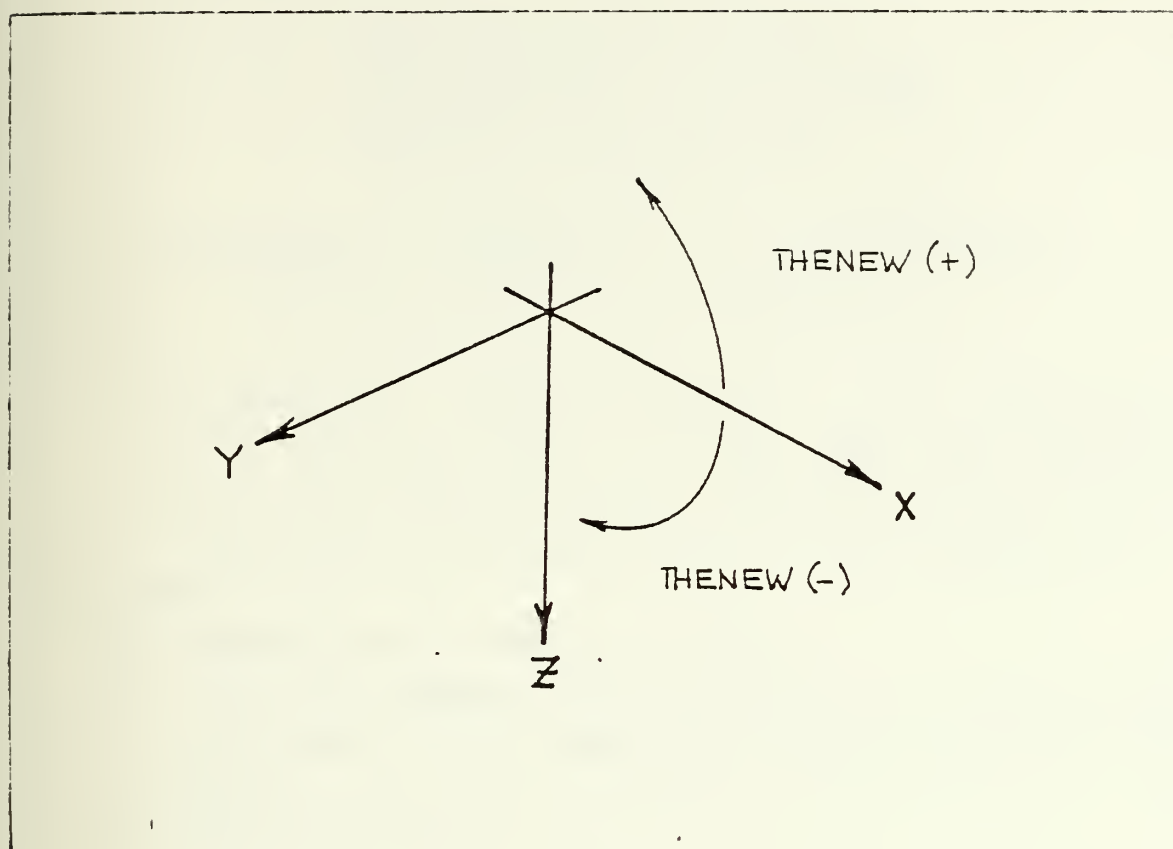


Figure 5.5 Sense for Pitch Above/Below the Horizon.

the horizontal plane (horizon). Positive is defined as nose above the horizon. THENEW is measured in radians. See figure 5.5. The roll angle PHINEW defines how much the right hand wing of the "aircraft" is rotated above or below the horizon. PHINEW is a function of PSINEW and is related through the turn rate of the aircraft. The equation

$$\text{TRNRAT} = (\text{PSINEW} - \text{PSIOLD}) / \text{DELTEE}$$

defines turn rate as the heading change that took place going from "old" to "temp" over the period DELTEE. The factor 9.7958 that appears in equations

```
IF (PSINEW.GT.0.) PHINEW=
*   ATAN (ABS (TRNRAT*VELOT/9.7958))
IF (PSINEW.LT.0.) PHINEW=
*   -ATAN (ABS (TRNRAT*VELOT/9.7958))
```

is derived in appendix B. The logic statement

```
IF (PSINEW.EQ.PSIOLD) PHINEW=PHIOLD
```

accounts for constant heading. It is written in this fashion to smooth the roll angle and therefore roll rate through the "new" waypoint. This will be clarified in the comments on the next subroutine.

11. ROTRAT

- a. expanded name: Rotational Rates
- b. function: Compute the orthogonal rotational rates that occurred as a result of the move from "old" to "temp".
- c. equations/calls:

```
PSIDOT=(PSINEW-PSIOLD)/DELTEE
THEDOT=(THENEW-THEOLD)/DELTEE
PHIDOT=(PHINEW-PHIOLD)/DELTEE
```

- d. comments: See figure 5.6 for rotation sense.

12. ACIRM1

- a. expanded name: Aircraft Transformation Matrix #1
- b. function: Transform the velocity and acceleration components referenced to the earth-fixed coordinate system (x pointing East and z pointing down) into components referenced to coordinate system with x

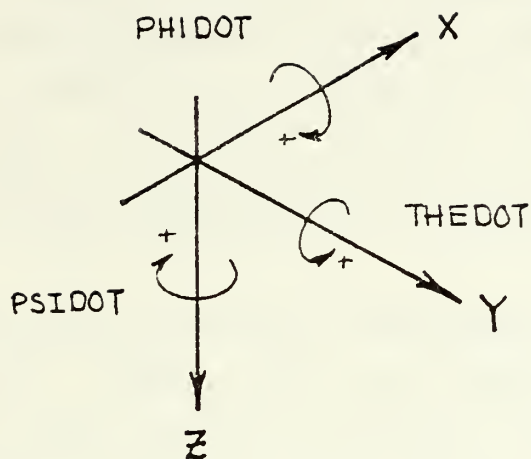


Figure 5.6 Definition of Positive Rotations.

pointing along the nose of the aircraft (but still in the horizontal plane) and z pointing down.

c. equations/calls:

```

ANG=PSINEW
U2= COS (ANG) *U1+SIN (ANG) *V1
V2=-SIN (ANG) *U1+COS (ANG) *V1
W2=W1
UD2= COS (ANG) *UD1+SIN (ANG) *VD1
VD2=-SIN (ANG) *UD1+COS (ANG) *VD1
WD2=WD1

```


d. comments: See appendix C for a development of all three transformation matrices, ACTRM1, ACTRM2, and ACTRM3. The requirement for the matrices stems from the requirement to transform the earth-fixed coordinate system velocity and acceleration components into aircraft-fixed coordinate velocity and acceleration components. Because the rotations being dealt with are discrete rather than differential, the order of the transformation matrices is fixed. This is also demonstrated in appendix C.

13. ACTRM2

- a. expanded name: Aircraft Transformation Matrix #2
- b. function: Carry out the second transformation whereby the x-axis is rotated up/down out of the horizontal plane to go through the nose of the aircraft.
- c. equations/calls:

```
ANG=THENEW
U3=COS (ANG) *U2-SIN (ANG) *W2
V3=V2
W3=SIN (ANG) *U2+COS (ANG) *W2
UD3=COS (ANG) *UD2-SIN (ANG) *WD2
VD3=VD2
WD3=SIN (ANG) *UD2+COS (ANG) *WD2
```

d. comments: See appendix C.

14. ACTRM3

- a. expanded name: Aircraft Transformation Matrix #3
- b. function: Carry out the third transformation whereby the y-axis is rotated up/down out of the horizontal plane to go through the right wing (as referenced from the cockpit).
- c. equations/calls:


```

ANG=PHIN EW
UAC=U3
VAC= COS (ANG)*V3+SIN (ANG)*W3
WAC=-SIN (ANG)*V3+COS (ANG)*W3
UDOTAC=UD3
VDOTAC= COS (ANG)*VD3+SIN (ANG)*WD3
WDOTAC=-SIN (ANG)*VD3+COS (ANG)*WD3

```

d. comments: See appendix C.

15. ACRVEL

- a. expanded name: Aircraft Rotational Velocities
- b. function: Convert (transform) the rotational velocities from the earth-fixed coordinate system to the aircraft-fixed coordinate system.
- c. equations/calls:

```

ANG1=THEN EW
ANG2=PHIN EW
P=PHIDOT-SIN (ANG1)*PSIDOT
Q= COS (ANG2)*THEDOT
*   COS (ANG1)*SIN (ANG2)*PSIDOT
R=-SIN (ANG2)*THEDOT
*   COS (ANG1)*COS (ANG2)*PSIDOT

```

d. comments: See figure 5.7 for sense. See appendix C for derivation.

16. ACACC

- a. expanded name: Aircraft Accelerations
- b. function: Sum up the linear and rotational accelerations imposed on the aircraft.
- c. equations/calls:

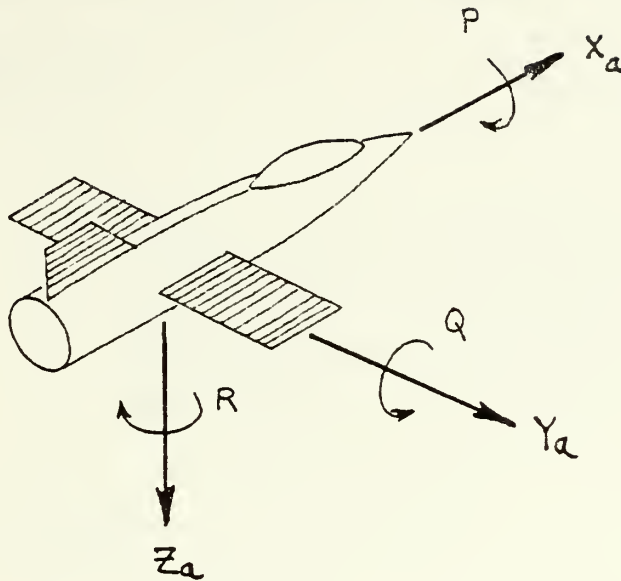


Figure 5.7 Aircraft Rotational Velocities.

$$AX = UDOTAC - VAC * R + WAC * Q$$

$$AY = VDOTAC + UAC * R - WAC * P$$

$$AZ = WDOTAC - UAC * Q + VAC * P$$

d. comments: Ncne.

17. ACLMIS

a. expanded name: Aircraft Limitations

b. function: This is the key subroutine of the trajectory package. It calls for computation of the forces imposed on the aircraft and compares the forces to allowable values. If the forces are too great, it initiates a selective reduction in the earth-fixed acceleration components until the

forces on the aircraft fall within limitations. If the forces are less than allowable values it initiates a selective increase in earth-fixed acceleration components until the maximum allowable is reached.

c. equations/calls:

```
CALL THRUST
      IF (AN.LT.ANMAX.AND.FWDACC.LT.THMAX)
*      CALL BOOSTR
      IF (AN.GT.ANMAX.OR.FWDACC.GT.THMAX)
*      CALL REDUCR
```

d. comments: Ncne.

18. XCHNGE

a. expanded name: Exchange

b. function: Once the performance of the aircraft has been maximized at the point "temp", "temp" becomes accepted as a point along the trace toward waypoint "new". "Old" is moved up to "temp".

c. equations/calls:

```
XOLD=XTEMP
YOLD=YTEMP
ZOLD=ZTEMP
UOLD=UTEMP
VOID=VTEMP
WOLD=WTEMP
PSICLD=PSINEW
THECLD=THENEW
PHICLD=PHINEW
VEICO=VELOT
```


d. comments: This is, in effect, the movement forward of the "aircraft" to "temp", the next increment toward waypoint "new".

19. WAYPNT

- a. expanded name: Waypoint
- b. function: Load the accepted values into vectors for use elsewhere in the non-BIGCAL portion of IBMPIP.
- c. equations/calls:

```
WX (ICT)=XOLD
WY (ICT)=YOLD
WZ (ICT)=ZOLD
WU (ICT)=UOLD
WV (ICT)=VOLD
WW (ICT)=WOLD
WH (ICT)=- PSINew
WP (ICT)= THENew
WR (ICT)=- PHINew
WG (ICT)=AN
T (ICT)=T (ICT-1) +DELTEE
```

- d. comments: The minus signs appear in WH(ICT)= and WR(ICT)= to account for transformation back into the original map coordinate system which has z pointing up.

20. FILE

- a. expanded name: Write to file
- b. function: Immediately write all the appropriate information to disk file for future use
- c. equations/calls:

```
IF (XOLD.NE.XNEW.OR.YOLD.NE.YNEW.OR.ZOLD.NE.ZNEW)
*   GOTO 10
```



```

WRITE(11,100) T(ICT),WX(ICT),WY(ICT),WZ(ICT),
*
*      WU(ICT),WV(ICT),WW(ICT),
*      WH(ICT),WP(ICT),WR(ICT),
*      WG(ICT),LTR
RETURN
10      CONTINUE
      LTR=0
      WRITE(11,100) T(ICT),WX(ICT),WY(ICT),WZ(ICT),
*
*      WU(ICT),WV(ICT),WW(ICT),
*      WH(ICT),WP(ICT),WR(ICT),
*      WG(ICT),LTR
100    FORMAT(11(F10.2,1X),I3)

```

d. comments: The if statement is the current method of "breaking out" of BIGCAL when the program arrives at the waypoint "new". To aid in reading the data files the control variable LTR is set to zero for all the interim points on the curve. LTR is a notification by the user of arrival at certain key points in the game scenario and are described elsewhere.

21. GRAFIX

- a. expanded name: Graphics subroutine
- b. function: Use the WX(ICNT) and WY(ICNT) vector values to plot the trace of the flightpath on the graphics screen.
- c. equations/calls:

```

PX1=WX(ICT-1)
PY1=WY(ICT-1)
PX2=WX(ICT)
PY2=WY(ICT)
CALL WIN(MINX,MAXX,13000.,12000.,0)
CALL GBVECT(0.,PX1,PY1)

```



```
CALL GBVECT(1.,PX2,PY2)
```

```
CALL GSFRCE
```

d. comments: The equations

```
PX1=WX(ICT-1)
```

```
PY1=WY(ICT-1)
```

```
PX2=WX(ICT)
```

```
PY2=WY(ICT)
```

establish the beginning and end points of the draw.

The call to the subroutine WIN (window)

```
CALL WIN(MINX,MAXX,18000.,12000.,0)
```

reaches into non-BIGCAL IBMPIP. For here it is simply described as a definer of the portion of the graphics screen on which the drawing is to take place. The calls

```
CALL GBVECT(0.,PX1,PY1)
```

```
CALL GBVECT(1.,PX2,PY2)
```

are calls to the graphics package GRAF 77, the IBM graphics package at Naval Postgraduate School. GBVECT passing a (0.) interprets as "Move to the following coordinates.". GBVECT passing a (1.) interprets as "Draw to the following coordinates". The final call

```
CALL GSFRCE
```

forces the actual updating of the graphics screen with the move and draw just called for.

C. THIRD LAYER SUBROUTINES

The following subroutines are accessed through SIGCAL. Their overall function is to compute the forces on the "aircraft" based on the accelerations in the "aircraft's" coordinate system. Once the forces are established, a forth level subroutine, which will be included here for continuity, re-drives the higher level subroutines to iterate (ultimately) the forces to within limitations.

1. THRUST

- a. expanded name: Thrust and g-load
- b. function: Establish the forward acceleration and g-load on the airframe.
- c. equations/calls:

```
ANG=THENEW
THRQD=AX/GEE
AN=SQRT (AY**2+AZ**2) /GEE
SIGMA=(1+DTDH*ZTEMP/TO) **4.257
RHOALT=RHOSSL*SIGMA
CL=(2.*AN*WL)/(RHOALT*VELOT**2)
DRAG=AN*((CDO+CDK*CL**2)/CL)
FWDACC=SIN (ANG) +THRQD+DRAG
```

- d. comments: The equation

```
ANG=THENEW
```

establishes the "aircraft's" attitude above or below the horizon and is necessary for computing the direction through which gravity forces act upon the "aircraft". The following equations respectively

```
THRQD=AX/GEE
AN=SQRT (AY**2+AZ**2) /GEE
```


compute the longitudinal non-dimensionalized acceleration based on the user's desired move and the composite non-dimensionalized acceleration perpendicular to the longitudinal axis. The next equation

$$\text{SIGMA} = (1 + \text{DTDH} * \text{ZTEMP} / \text{TO}) ** 4.257$$

is the density ratio (altitude to sea-level) and is a function of standard temperature lapse rate and altitude. It is derived in appendix B. The density at altitude is then

$$\text{RHOALT} = \text{RHOSSL} * \text{SIGMA}$$

the density at sea-level times the given ratio. The density, velocity, non-dimensional g-load and wing loading (found in BLOCK DATA) are

$$\text{CL} = (2. * \text{AN} * \text{WL}) / (\text{RHOALT} * \text{VELOT} ** 2)$$

fed into the equation for coefficient of lift. Drag is a function of the coefficient and is given as:

$$\text{DRAG} = \text{AN} * ((\text{CDO} + \text{CDK} * \text{CL} ** 2) / \text{CL})$$

The components gravity, forward acceleration (required) and the

$$\text{FWDACC} = \text{SIN}(\text{ANG}) + \text{THRQD} + \text{DRAG}$$

resulting drag on the airframe are summed to give the total forward acceleration on the "aircraft". (The acceleration will be negative if user "slows down".)

2. BOOSTR

- a. expanded name: Performance booster
- b. function: Increase the "aircraft" performance up to the allowable maximum thrust or g-load, whichever is smaller.

c. equations/calls:

```

      MCONT=1
5      CONTINUE
      I=0
      IF (AG1.EQ.DELX.OR.AG1.EQ.0.) GOTO 10
          IF (ABS (AG1) .LT. ABS (AH1) .AND. ABS (AG1) .LT.
*          ABS (AI1)) I=1
              IF (I.NE.1) GOTO 10
                  UDOT=1.02*UDOT
10     CONTINUE
      IF (AH1.EQ.DELY.OR.AH1.EQ.0.) GOTO 20
          IF (ABS (AH1) .LT. ABS (AG1) .AND. ABS (AH1) .LT.
*          ABS (AI1)) I=2
              IF (I.NE.2) GOTO 20
                  VDOT=1.02*VDOT
20     CONTINUE
      IF (AI1.EQ.DELZ.OR.AI1.EQ.0.) GOTO 30
          IF (ABS (AI1) .LT. ABS (AG1) .AND. ABS (AI1) .LT.
*          ABS (AH1)) I=3
              IF (I.NE.3) GOTO 30
                  WDOT=1.02*WDOT
30     CONTINUE
      IF (AG1.EQ.DELX.OR.AG1.EQ.0.) GOTO 40
          IF (ABS (AG1) .EQ. ABS (AI1) .AND. ABS (AG1) .LT.
*          ABS (AH1)) I=4
              IF (I.NE.4) GOTO 40
                  UDOT=1.02*UDOT
                  WDOT=1.02*WDOT
40     CONTINUE
      IF (AH1.EQ.DELY.OR.AH1.EQ.0.) GOTO 50
          IF (ABS (AG1) .EQ. ABS (AH1) .AND. ABS (AG1) .LT.
*          ABS (AI1)) I=5
              IF (I.NE.5) GOTO 50
```



```

                                UDOT=1.02*UDOT
                                VDOT=1.02*VDOT
50      CONTINUE
        IF (AI1.EQ.DELX.OR.AI1.EQ.0.) GOTO 10
            IF (ABS(AH1).EQ.ABS(AI1).AND.ABS(AH1).LT.
*          ABS(AG1)) I=0
                IF(I.NE.3) GOTO 60
                    VDOT=1.02*VDOT
                    WDOT=1.02*WDOT
60      CONTINUE
        CALL SMLCAL
        CALL THRUST
        IF (AN.GE.ANMAX.OR.FWDACC.GE.IHMAX) GOTO 70
            GOTO 5
70      CONTINUE
        IF (AN.GT.ANMAX) AN=ANMAX
        IF (FWDACC.GT.IHMAX) FWDACC=IHMAX

```

d. comments: The logic statements

```

        IF (AG1.EQ.DELX.OR.AG1.EQ.0.) GOTO 10
        IF (AH1.EQ.DELY.OR.AH1.EQ.0.) GOTO 20
        IF (AI1.EQ.DELZ.OR.AI1.EQ.0.) GOTO 30
        IF (AG1.EQ.DELX.OR.AG1.EQ.0.) GOTO 40
        IF (AH1.EQ.DELY.OR.AH1.EQ.0.) GOTO 50
        IF (AI1.EQ.DELZ.OR.AI1.EQ.0.) GOTO 60

```

avoid iterating on orthogonal directions that are defined as zero by user input and to turn off iteration when then "aircraft" has arrived at the destination (in that coordinate plane). The logic equations that follow choose the direction

```

                                IF (ABS(AG1).LT.ABS(AH1).AND.ABS(AG1).LT.
*          ABS(AI1)) I=1
                                IF (ABS(AH1).LT.ABS(AG1).AND.ABS(AH1).LT.

```



```

*      ABS(AI1)) I=2
      IF (ABS(AI1).LT.ABS(AG1).AND.ABS(AI1).LI.
*      ABS(AH1)) I=3
      IF (ABS(AG1).EQ.ABS(AI1).AND.ABS(AG1).LI.
*      ABS(AH1)) I=4
      IF (ABS(AG1).EQ.ABS(AH1).AND.ABS(AG1).LI.
*      ABS(AI1)) I=5
      IF (ABS(AH1).EQ.ABS(AI1).AND.ABS(AH1).LI.
*      ABS(AG1)) I=6

```

in which the iteration is to take place. The selection criteria is the coordinate plane in which there is the least distance left to go to the destination. Thus for small altitude changes, the "aircraft" will "pop-up" to the desired altitude as soon as physically possible. In other situations the "aircraft" will turn in shortest direction first. Finally, the acceleration (in map coordinates) in the selected coordinate plane is increased in magnitude

```
UDOT=1.02*UDOT
```

```
VDOT=1.02*VDOT
```

```
WDOT=1.02*WDOT
```

until performance limitations are reached.

3. REDUCR

- a. expanded name: Demanded performance reducer
- b. function: For a user defined waypoint that exceeds the allowable maximum thrust or g-load, whichever is smaller, reduce the performance demand.
- c. equations/calls:

```

MCONT=1
5      CONTINUE
      VELON=VELON-DELTV
      A=VSTALL+2.*DELTV

```



```

      IF (VELON.GE.A) GOTO 10
      ANG=ACOS (AG1/AJ2)
      AJ2=1.1*AJ2
      AG1=COS (ANG) *AJ2
      AH1=SIN (ANG) *AJ2
      VELON=V1
10    CONTINUE
      N=0
      CALL MVELO S
      CALL MVDOTS
      CALL SMLCAL
      CALL THRUST
      IF (AN.LE.ANMAX.AND.FWDACC.LE.THMAX) GOTO 20
      GOTO 5

```

d. comments: The tact adopted is different from the one chosen for BCOSTR. The "aircraft" in moving from "old" to "new" has a radius of curvature. When the performance limitations of the "aircraft" are exceeded, this means the "aircraft" cannot fly with a small enough radius of curvature to converge on the waypoint "new". The two available means of meeting the demand are to decrease the "aircraft's" deliverable radius of curvature by reducing the destination ("new") velocity or increase the demanded radius of curvature to something the "aircraft" can accomplish by moving the point "new" further away from the point "old". Frequently a small change in velocity is sufficient, therefore

VELON=VELON-DELTV

is chosen as the first iteration. DELTV is a velocity increment found in BLOCK DATA and

represents approximately five knots airspeed in meters per second. This value was chosen as representative of an attack pilot's maximum precision in airspeed control. There is a lower bound to which airspeed may be reduced, and that is the stall limitation of the aircraft. When the "aircraft" is

```
A=VSTALL+2.*DELT  
IF(VELON.GE.A) GO TO 10
```

close to this limit, the second portion of the subroutine is invoked

```
ANG=ACOS(AG1/AJ2)  
AJ2=1.1*AJ2  
AG1=COS(ANG)*AJ2  
AH1=SIN(ANG)*AJ2  
VELON=V1
```

to move the point "new" further out. Note the final equation; the view is adopted that an attack pilot will not wish to be at a minimum energy anywhere in the approach phase. Therefore the originally selected airspeed is reset into the iteration.

In summation, this approach was taken in preference to merely setting demanded performance equal to maximum deliverable performance to avoid the unrealistic scenario depicted in figure 5.8

4. SMLCAL

- a. expanded name: Small call
- b. function: This forth level subroutine was written as a programmer convenience to drive the iterations.
- c. equations/calls:

```
CALL DELVEC  
CALL TMPPNT
```

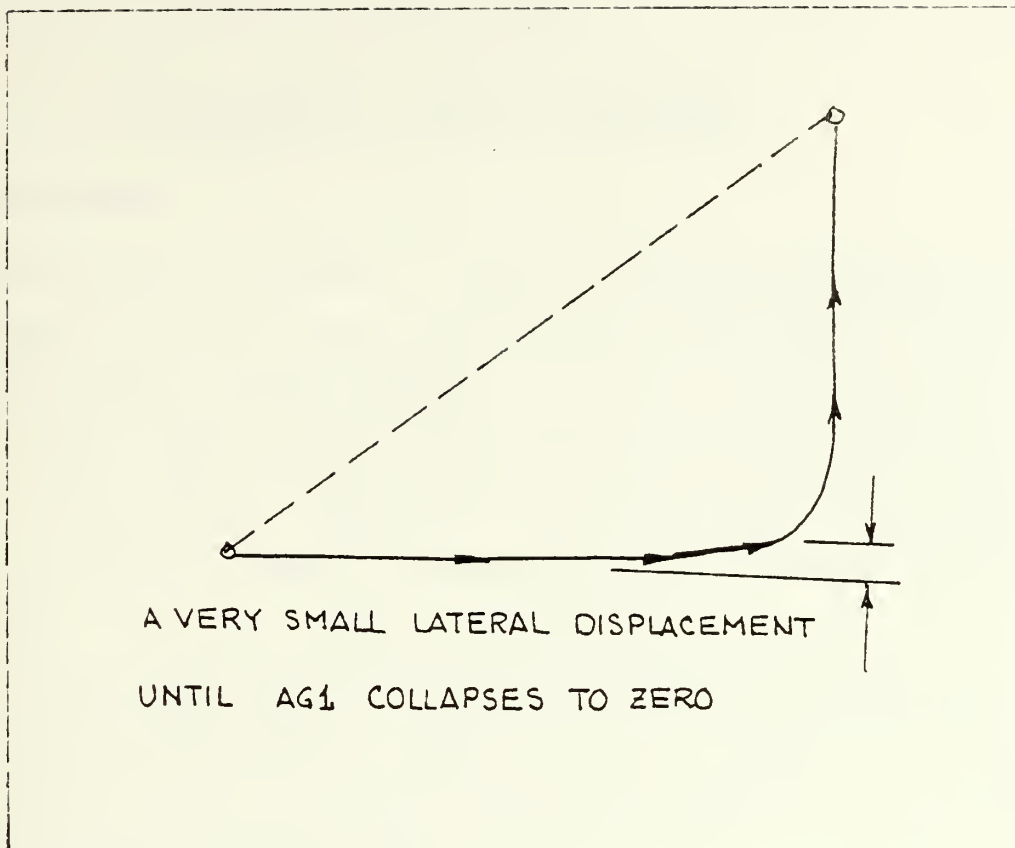



Figure 5.8 Unrealistic Scenario.

```
CALL TMPVEL
CALL INETRM
CALL ANGLES
CALL ROTRAT
CALL ACTRM1
CALL ACTRM2
CALL ACTRM3
CALL ACRVEL
CALL ACACC
```

d. comments: As can be seen the subroutines are those previously described.

VI. NON-BIGCAL IBMPIP SUBROUTINES

A. INTRODUCTION

For the sake of brevity, the only subroutine addressed in the section below is the one that was drastically altered to accomodate the BIGCAL package. The rest of the subroutines that are not mentioned may be found in the listing at the end of the thesis. They will have changes from the Johns [Ref. 1] version of IBMPIP but will also have extensive numbers of comment cards to describe to key points of each subroutine.

The final section provides a consolidated listing of the graphics calls along with a short description of their use. For follow-on alterations to this version of IBMPIP the user should consult [Ref. 2] for complete description of the available graphics subroutines.

B. SUBROUTINES

1. ERRCHK

- a. expanded name: Error checking
- b. function: Check the user defined "new" waypoint for game rule violations or user commands before entering into the BIGCAL subroutines.
- c. equations/calls:

```
DX=TARGX-X 1
DY=TARGY-Y 1
DIST=SQRT (DX**2+DY**2)
IF (DIST.LE.POPMIN) GOTO 10
    IF (Z1.GT.APPMAX) Z1=APPMAX
    IF (Z1.LT.HTMIN) Z1=HTMIN
```



```

10      CONTINUE
      IF (LTR.NE.65.OR.LIR2.NE.65) GOTO 20
          IF (Z1.LT.BAALT) Z1=BAALT
20      CONTINUE
      IF (LTR.NE.66.OR.LIR2.NE.66) GOTO 60
          IF (Z1.LT.BRMIN) Z1=BRMIN
          IF (Z1.GT.BRMAX) Z1=BRMAX
          IF (DIST.LE.BRRMAX.AND.DIST.GE.BRRMIN)

```

GOTO 30

```

          IF (DIST.GT.BRRMAX) IERR=9
          IF (DIST.LT.BRRMIN) IERR=13
          RETURN

```

```

30      CONTINUE
      TGTHDG=ATAN2(DY,DX)*RTD
      IF (TGTHDG.LT.0.) TGTHDG=TGTHDG+360.
      DX=X1-WX (ICT)
      DY=Y1-WY (ICT)
      ACHDG=ATAN2(DY,DX)*RTD
      IF (ACHDG.LT.0.) ACHDG=ACHDG+360.
      HDGLMT=ABS(TGTHDG-ACHDG)
      IF (HDGLMT.LE.5.) GOTO 40
          IERR=10
          RETURN

```

```

40      CONTINUE
      DIST=SQRT(DX**2+DY**2)
      SPEED=SQRT(WU (ICT) **2+WV (ICT) **2+WW (ICT) **2)
      DT=DIST/SPEED
      IF (DT.GE.2.33) GOTO 50
          IERR=10
          RETURN

```

```

50      CONTINUE

```

```

60      CONTINUE

```


d. comments: The first equations establish the distance from

```
DX=TARGX-X 1
DY=TARGY-Y 1
DIST=SQRT(DX**2+DY**2)
```

the waypoint "new" to the target. The next group of equations

```
IF (DIST.LE.POPMIN) continue
    IF (Z1.GT.APPMAX) Z1=APPMAX
    IF (Z1.LT.HTMIN) Z1=HTMIN
CONTINUE
```

look at the user input altitude. If the user exceeds either the maximum or minimum inbound altitudes, the input altitude is automatically set to the limiting parameter. APPMAX, HTMIN, and POPMIN are found in BLOCK DATA. The next group of lines satisfy the game parameter that the "aircraft" must be "popped-up" to a given altitude before aim at the target can be taken. LTR is the code passed from

```
IF (LTR.NE.65.OR.LTR2.NE.65) continue
    IF (Z1.LT.BAALT) Z1=BAALT
CONTINUE
```

the graphics subroutines signifying that the bomb aim line is desired. LTR2 is a back-up code passed from the same subroutines and allows the user to change his/her mind while still entering data. LTR is generated upon entry of the X-Y point on the graphics screen. LTR2 is generated upon entry of the Z-velocity point on the graphics screen. BAALT is the mnemonic for bomb aim altitude and is found in BLOCK DATA. The following statements all relate to

the bomb release point. If bomb release is not called for;

```
IF(LTR.NE.66.OR.LTR2.NE.66) GOTO 60
```

skips over all the rest and returns to MAIN. If at bomb release the altitude is too high or low,

```
IF(Z1.LT.BRMIN) Z1=BRMIN
```

```
IF(Z1.GT.BRMAX) Z1=BRMAX
```

set the altitude to the minimum/maximum allowable. BRMIN is the mnemonic for minimum bomb release altitude, BRMAX is similar, and both are defined in BLOCK DATA. Because the flight path is now curvilinear, there is no convenient way to correct for an input that violates minimum or maximum bomb release range (from target). Error codes are generated that inform the user that the just-entered point will have to be moved. BRMAX and BRMIN are found in BLOCK DATA.

```
IF(DIST.LE.BRMAX.AND.DIST.GE.BRMIN) continue
```

```
IF(DIST.GT.BRMAX) IERR=9
```

```
IF(DIST.LT.BRMIN) IERR=13
```

```
RETURN
```

```
CONTINUE
```

The following group of statements compare the heading from the aimpoint to the target with the aircraft heading to the target.

```
TGTHDG=ATAN2(DY,DX)*RTD
```

```
IF(TGTHDG.LT.0.) TGTHDG=TGTHDG+360.
```

```
DX=X1-WX(ICT)
```

```
DY=Y1-WY(ICT)
```

```
ACHDG=ATAN2(DY,DX)*RTD
```

```
IF(ACHDG.LT.0.) ACHDG=ACHDG+360.
```



```

HDGLMT=ABS(TGTHDG-ACHDG)
IF(HDGLMT.LE.5.) continue
    IERR=10
    RETURN
CONTINUE

```

If the "aircraft" heading is not within five degrees of the aimline the user is informed, and the point must be user moved. RTD is the mnemonic for radians to degrees and is found in BLOCK DATA. If the aircraft does not travel down the aimline for a minimum of 2.33 seconds

```

DIST=SQRT(DX**2+DY**2)
SPEED=SQRT(WU(ICT)**2+WV(ICT)**2+WW(ICT)**2)
DT=DIST/SPEED
IF(DT.GE.2.33) GOTO continue
    IERR=11
    RETURN
CONTINUE

```

the user is informed, and must make the correction.

C. GRAPHIC CALLS

DSINIT

function: Initialise both of the 3277 screens.

DSTERM

function: Terminates dual screen graphics activity.

GASVIE(aa,bb)

function: Defines a viewport, or a smaller portion of the total graphics screen available. Allows subdividing the total screen into several smaller pictures.

GASWIN(aa,bb)

function: Takes the data being supplied by the user's equations and converts it into absolute graphic screen coordinates in preparation for drawing. Takes care of proportionality.

GAXCTI(aa,bb,cc,dd,ee)

function: Convert user supplied character strings into integer vectors.

GAXITC(aa,bb,cc,dd,ee)

function: Convert one or more integers into characters in preparation for writing onto the graphics screen.

GBCHAR(aa,bb,cc,dd,ee,ff)

function: Invokes two lower-level graphics calls to write characters on the graphics screen in the defined font.

GBGARC(aa,bb,cc,dd,ee,ff,gg)

function: Invokes two lower-level graphics call to generate an arc of either a circle or an ellipse (can also be the whole circle, etc).

GBGDOI(aa,bb,cc)

function: Invokes two lower-level subroutines to draw a filled-in dot.

GBRXYC(aa,bb,cc)

function: Read one graphic data point using the cursor cross-hairs.

GBVECT(aa,bb,cc)

function: Generates a "move-to" or a "draw-to" command, depending on whether (aa) is zero or non-zero (real number).

GSERSE

function: Erase the graphics screen.

GSFRCE

function: Force an immediate update of the graphics screen with new data.

GSLI(aa)

function: Sets the line type to be used for drawing (dot, dashed, etc.).

VII. STATUS OF PROPOSED IBMPIP AND SUMMARY

Despite best efforts, the proposed version is not ready for implementation. The flaws revealed by extensive testing, center around the two iterative subroutines and are felt to be solvable. In particular, subroutine MVDOTS does not account for the following situation, see figure 7.1:

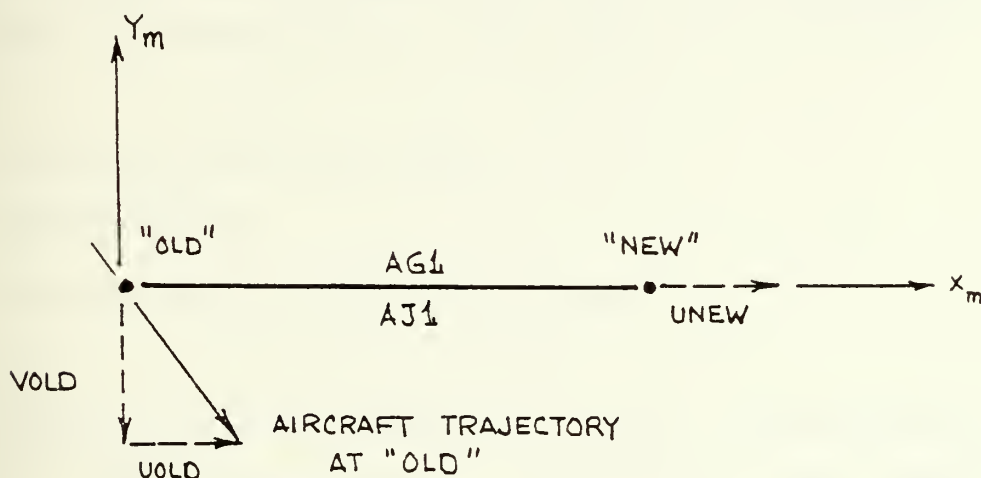


Figure 7.1 A Situation Not Addressed By Subroutine ACACC.

$$ACC = (VELON**2 - VELOO**2) / (2 * AJ1)$$

$$UDOT = (AG1 / AJ1) * ACC = 1. * ACC = ACC$$

$$VDOT = (0. / AJ1) * ACC = 0.$$

We see that by the model, $VDOT=0$. Realistically, this will not be so. There is a $VOLD$ and the "aircraft" will move some Y distance below "old". Let that distance be named $Y(unknown)$ or $YUNK$. $VDOT$ then will actually be:

$$VDOT = VOLD * 2 / (2 * YUNK) \text{ -- or some positive value.}$$

The best estimate at an approach would be either to guess a temporary distance based upon the ratio;

$$(VOLD/UOLD) * AG1$$

or to guess a temporary distance based upon a linear guess of the elapsed time between "old" and "new". Let linear time estimate be represented by LTE , then:

$$LTE = ABS((UNEW - UOLD) / UDOT)$$

$$YUNK = VOLD * LTE$$

$$VDOT = -VOLD / (2 * ((UNEW - UOLD) / UDOT))$$

Another area of concern is in the definition of the angles in subroutine ANGLES. Artificially large effects are felt in the angles $PSINEW$ and $PHINEW$ if $DELX$ or $DELY$ collapse to zero before the "aircraft" arrives at "new" in all three orthogonal planes. Additionally, large effects are produced in the last DEL -movement to "new", causing a hang-up in the $BOCSTR/REDUCR$ subroutines. Because the current game restricts altitude changes to approximately one magnitude less than either X or Y changes, the effect by $THENEW$ (θ) is roughly one magnitude less throughout. To pave the way for future efforts to smooth out the angle changes between "old" and "new", the variable counter $MCCNT$ was added to $COMMON/ABE5/$. $MCCNT$ is set from 0 to 1 by activation of either $BOCSTR$ or $REDUCR$ and switched back to 0

upon exit from either iterative loop. As numerical data are written to test files, the addition of MCONT as another column allows differentiation of first-guess values from the BOOSTR/REDUCR generated values.

The question was asked, "If the angles (and therefore rotational rates and therefore angular acceleration) components are so troublesome, how necessary are they to the accuracy of the output?". An analysis of the individual acceleration components in subroutine ACACC revealed that omission of the angular accelerations could lead to as much as a 50% difference in the magnitude of the net acceleration in any one of the orthogonal directions imposed on the aircraft. The answer then, is that smooth generation of angles is necessary if the modelled trajectory is to be considered faithful to a real flightpath.

In summation, a proposed IBMPIP improvement has been introduced and discussed. It is substantially more user friendly. It employs increased use of specialized graphics from [Ref. 2] and when completely debugged, will provide a far more accurate flightpath trace with less user interaction necessary.

APPENDIX A
USER'S MANUAL FOR CURRENT IBMPIP.

USER'S MANUAL

AE 3251
AIRCRAFT COMBAT SURVIVABILITY

A STUDY
of
AIRCRAFT ATTRITION
in a
HOSTILE AAA AND SAM ENVIRONMENT

NAVAL POSTGRADUATE SCHOOL
MCNTEREY, CALIFORNIA
WINTER QUARTER, 1984

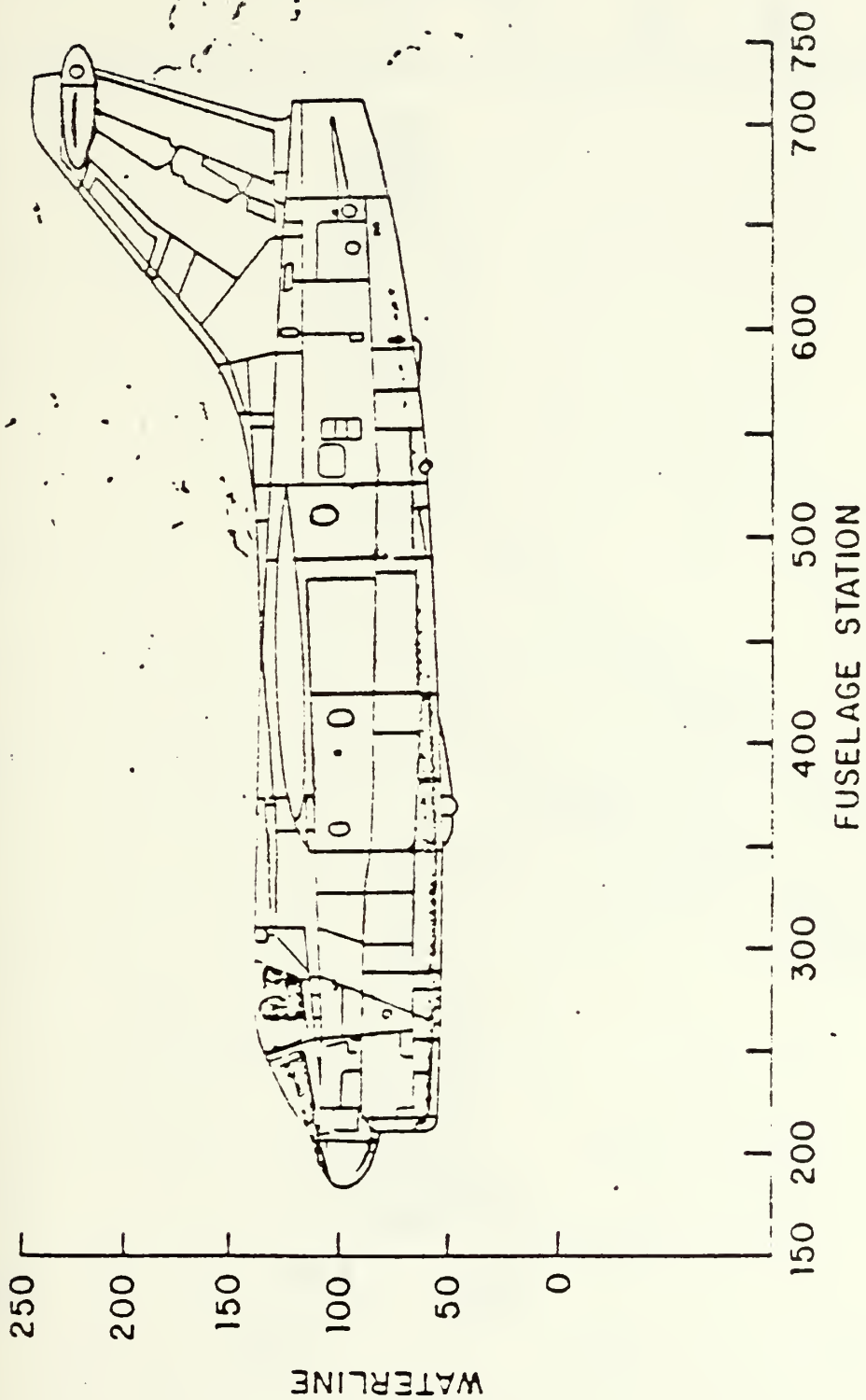
I. INTRODUCTION

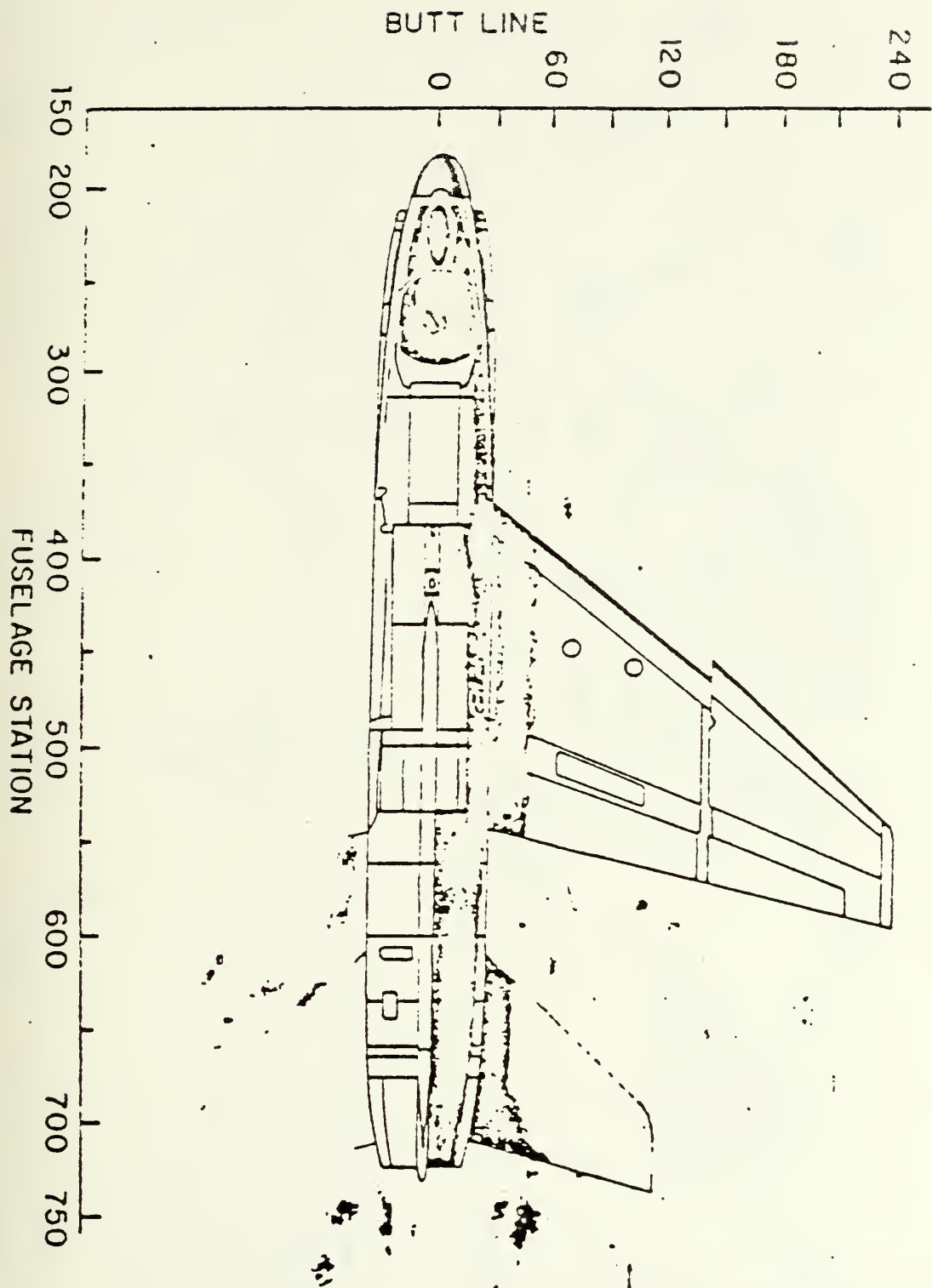
This aircraft attrition study is designed to present the student with an opportunity to see first-hand how the survivability of an aircraft can be evaluated in a given combat scenario. The methods employed in this study are those used by both industry and government when making decisions in the survivability analysis and design of an aircraft system. In this study, one computer program named P001 (the AFATL Antiaircraft Artillery Simulation Computer Program) will be used to simulate the flight of a typical Naval attack aircraft through a hostile antiaircraft artillery (AAA) environment and to compute the aircraft probability of survival. Another computer program, MICE II (the Missile Intercept Capability Evaluation Program), will be used to compute the survivability of the aircraft on the same flight path against a typical short-to-medium range surface-to-air missile (SAM).

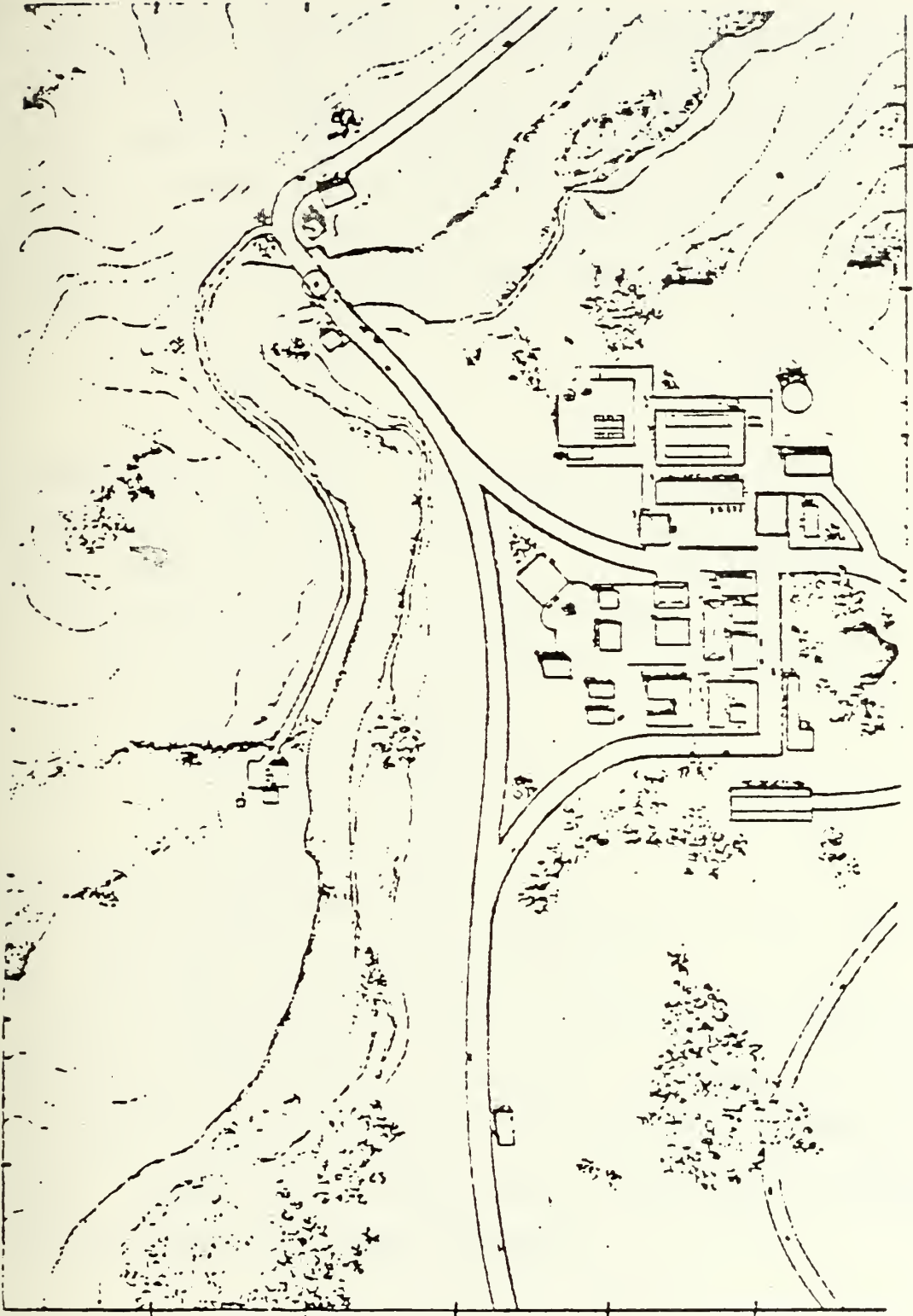
The DOD specifies the use of P001 and MICE in all nonnuclear survivability assessments in MIL-STD-2069, REQUIREMENTS FOR AIRCRAFT NONNUCLEAR SURVIVABILITY PROGRAM, 24 August 1981.

II. PROBLEM DEFINITION

- A. You are going to conduct a survivability assessment of a familiar Naval aircraft, shown in Figs. 1 and 2, on a typical attack mission to destroy the bridge shown in Fig. 3.
- B. The class will be divided into groups of four (4), with two members in each group on the BLUE team and two members on the RED team.
- C. Each team will use P001 and MICE II to determine the survivability of the aircraft on the class problem scenario, as follows:
 1. Each team will generate a flight a flight path to the bridge according to the rules of the scenario given in Section IV. Keep this path a secret.
 2. Each team will also select the locations of six AAA emplacements and one SAM emplacement that will defend against air attack. Locate the weapons according to the order of battle given in Section IV. Keep these locations a secret also.
 3. Each team will conduct an attack against the other team in the following fashion: The two teams in each group will conduct a trade of offensive data files. That is; RED teams flight path will go up against the BLUE teams defensive positions and BLUE teams flight path will go up against RED teams defensive positions.







III. SCENARIO DESCRIPTION

A. This scenario is purely for instructional purposes and is not based on any actual or planned combat attack situation. The target site, order of battle, and flight path and weapon delivery parameter limits have been chosen only to provide guidelines for the class problem. As much realism has been introduced for the players as possible while retaining an unclassified scenario.

b. Your target is the bridge shown in Fig. 3 located at :

X -- 14,000 mtrs

Y --- 7,220 mtrs

Z ----- 20 mtrs

Heavy military supply traffic has been reported in this area. Your mission is to destroy this vital supply link.

C. The following order of battle has been gathered from intelligence reports of the target area:

SAM - one site in the vicinity of the target

AAA - (2) type 1 mode 1

(2) type 2 mode 1

(1) type 3 mode 3

(1) type 3 mode 4

(1) type 5 mode 3

****NOTE**** Gun types and their relationship to AAA and the SAM type will be discussed in class.

D. The SAM threat requires that the inbound approach to the target be made from the west at low level. A pop-up maneuver is required to visually identify the target followed by a dive bombing run to weapon release. The aircraft ordnance consists of MK82 500-pound Snakeeye bombs. Egress must be made either to the north or south, depending upon individual strategy.

E. The following list of scenario limitations to be used in the development of your strategy:

1. Aircraft speed limitations

- a. minimum (stall) ----- 90 m/s (175 knots)
- b. maximum, with bombs ----- 260 m/s (505 knots)
- c. maximum, bombs released -- 310 m/s (603 knots)

2. Aircraft altitude limitations

- a. prior to pop-up
 - 1). max --- 457 mtrs
 - 2). min ---- 60 mtrs
- b. after pop-up
 - 1). max -- 2050 mtrs
 - 2). min ---- 60 mtrs

3. Maneuvering -- will be limited by:

- a. thrust
- b. speed brake size
- c. aircraft "g" loading --
the shorter the leg distance
the greater the restriction on:
 - 1). amount of turn OR
 - 2). airspeed/altitude changes

****NOTE**** Remember the following rule of thumb -- The greater a change in altitude the closer together minimum and maximum allowable velocities get.

4. Pop-up maneuver

- a. max allow dist (from tgt) to pop-up -- 6000 mtr
- b. minimum acceptable pop-up altitude --- 1000 mtr

5. Ordnance delivery

- a. boresight dive/glide only type acceptable
 - b. dive limitations
 - 1) must last -- 2.33 sec (606 mtrs at 260 knots)
 - 2) heading ---- must be within 5 degrees
- **NOTE**** The closer the attack is pressed to the target, the more precisely placed

the bomb release point must be.

c. bomb release ranges

1) max allowable from tgt -- 1000 mtrs

2) min allowable from tgt --- 100 mtrs

d. bomb release altitude limitations

1) max allowable -- 2000 mtrs

2) min allowable --- 100 mtrs

F. Weapons emplacements

1. order of emplacement fixed by program and goes

(2) type 1 mode 1

(2) type 2 mode 1

(1) type 3 mode 3

(1) type 3 mode 4

a seventh type 5 mode 3 battery

is always fixed at location

X=12,800 mtr/Y=7,500 mtr (near bridge)

2. neither of the type 3 batteries may be placed closer than 3000 mtrs from bridge.

3. (1) type 5 SAM battery may be put anywhere east of the 6000 mtr X location

G. Locate the AAA & SAM batteries (constrained by the above conditions) to defend the bridge.

H. Develop a flight path originating at an entry point anywhere near (but not 'on') the western boundary, attack the bridge and exit the map anywhere near (but not 'on') the northern or southern boundaries.

I. May the force be with you.

IV. PROCEDURES FOR OPERATING IBMPIP

A. In gross terms, the following will take place:

1. a computer session with IbmPIP
2. submission of your data to (2) batch programs to determine your success
3. making a hard copy of your map from the Tektronix TEK618 terminal

B. The following equipment is required: a graphics station consisting of:

1. an IBM3277 alpha-numeric terminal
2. a Tektronix TEK618 graphics screen
3. an IBM cursor control joystick

Many will be familiar with the above from working with the 'DISSPLA' programs. Additional information will now be supplied to enable utilization for this interactive graphics program.

C. Preflight:

1. ensure graphics terminal switched on
2. joystick set up -- ensure all three (3) blue rocker switches are in the (NORMAL) position
3. the 3277 terminal should always be 'power on'
-- if not -- do so now and set screen intensity to your satisfaction

D. Program execution

Refer to Appendix A to link up to Aero Disk, run IBMPIP and generate a flight path. Appendix B shows the sequence for gun/missile positioning. The black arrow heads are user entered commands or responses.

The following notes amplify certain aspects of the procedure. Read the Appendices and refer to these notes.

GUNS: 0=DISK FILE; 1=TERMINAL; 2=PRESET

Do not choose DISK FILE until you have created one by utilizing TERMINAL to input some gun/missile locations. The locations entered at TERMINAL will on your disk as GUN LOC A1.

AIRCRAFT MILESTONE INPUT: 0=DISK 1=TERMINAL

For the same reason as above do not choose DISK until you have created a file by utilizing TERMINAL. Your flight path will be written on your disk as PTS LOC A1.

ERROR CHECKING: 0=NO CHECKING 1=CHECK FOR ERRORS

Recommend only using error checking when entering aircraft flight path at terminal. It will erroneously activate on good data points being read in from the data file you will create.

FLIGHT & GAME PARAMETERS 1=DEFAULT 2=USER INPUT

For class competition, select 1.

VILLAGE DRAWN 0=NOT DRAWN 1=DRAWN

User preference.

ENTER AIRCRAFT MILESTONES

Look at bottom right hand corner of the IBM screen. You will see MORE... This is a prompt for you to hit CLEAR key. When you do,

1. The border of the map on the graphics screen will flash and the red light on joystick control will illuminate.
2. You may or may not see cross hairs on your graphics screen. Move the joystick around and try to find it.

3. If the cross hair is too faint, turn CLOCKWISE the WRITE THROUGH INTENSITY knob under the graphics screen. Not too much....
4. If the cross hair is too intense it will wipe a bright band across map messing it up. Turn COUNTERCLOCKWISE the WRITE THROUGH INTENSITY knob, until it stops.
5. The cross hair may drift slowly without your moving the joystick. Two little slider switches, under and to the right of the joystick are used to null out this drift. Move the obvious one until unwanted motion stops.
6. To enter a point: Manipulate the joystick until the cross hairs are somewhere near the western border. The number keys on top of the keyboard are the ones used to enter a data point. Press any key greater than 4. A dot will appear and then the border around the altimeter/airspeed bar will flash, prompting you to move the crosshair over there. Do so.

The vertical cross hair is to be set on your desired airspeed (as read from the tick marks on the bottom), slow on the left, fast on the right. Altitude is determined by the horizontal cross hair. When satisfied, press a number key greater than 4. A dot will appear and the border will flash (if all goes well) prompting you to return to do the next point.

If all does not go well, the skinny rectangle to the right of the graphics screen flashes and you will get an error message. A list of all possible error messages is now provided. Read them and the suggested user responses.

***** AIRCRAFT *****

MAXIMUM BRAKING EXCEEDED

The aircraft unable to slow down to chosen speed -- increase speed.

MAX G EXCEEDED

Rare -- usually aircraft is not able to handle G-load of turn -- try slowing down a little.

MAX THRUST EXCEEDED

The airspeed chosen is too fast -- slow down a little.
NOTE When maneuvering near the target, the distance between MAX BRAKING and MAX THRUST can get to be small. If unable to find an airspeed between them, next try changing altitude. If still unable to fit between the limiters, you will have to back up and enter a new X-Y location and try again.

ALT TOO LOW

Self explanatory.

ALT TOO HIGH

Self explanatory

POP-UP TOO LOW

Self explanatory.

BMB DRCP LOW

Self explanatory.

BMB DROP HI

Self explanatory.

STALL

Rare -- try increasing airspeed.

HDG<5 DEG TO TGT

Must reenter the X-Y point. Do a better job of putting point on bomb aim line.

FINAL RUN<2.33 SEC

Must reenter the X-Y point -- put it further down the aimline toward the target.

NO HORIZONTAL MOTION

Rare You didn't move far enough away from last entered point. Reenter a new X-Y point.

MAX LIFT EXCEED

Extremely rare. Manuever too severe. Try adjusting speed downward or moving altitude closer to the last one entered.

***** GUNS *****

TOO CLOSE TO TGT

Type 3 guns not far enough out from bridge. -- Reenter.

***** MISSILE *****

X COORDINATE LESS THAN 6000'

Missile is too close to the western border. Move to east of 6000 mtr line.

If you make an error and do not throw away the X-Y point, the altimeter border will flash and you may make you airspeed/altitude corrections. If you choose to correct the X-Y point, the map border will flash and you must then go over and reenter the point, see the altimeter flash and

return to it to reenter the associated altitude/airspeed combination.

If all goes well, no error is made and the user proceeds. Once the second point is entered, the program will connect-the-dots. Observe Fig 4.

As you progress, your flight path may resemble Fig. 5. At this time note the bold rings on the map.

1. The outer ring is used for aircraft pop-up. Once inside the ring, you may at any time commence pop-up.
2. The inner ring is used for placing gun/missile batteries and is explained elsewhere.

Observe Fig. 6. In the artificial case presented here, the user knows where the guns are and is maneuvering around them to a thin spot in the defense. Up to point 11, the aircraft is maintained at a low altitude. Note where the dots are on the altimeter ber.

Observe Fig. 7. The user has popped up between points 12 and 13. Observe the lone altitude dot at about 1250 mtrs altitude. The user is ready to take aim as he is about to input point 13. Number 1 key is the aim button. Press it for both the X-Y point and the associated altitude. A dotted line will appear. This is the line you must be on to release the bombs.

Observe Fig. 8. Point 14 is entered using Number 2 key. Bombs Away. Press the Number 2 key for both the X-Y point and the associated altitude. Note the 3 dots at about 800 mtrs -- it took 3 trys to get the airspeed right. It is now time to separt Dodge City. Point 15 is entered using Number 4 key. Again , as always, do this for both the X-Y point and the associated altitude. Normal termination of the program occurs.

Summary of the special keys:

1=take aim

2=kcmb release

3=abnormal termination

4=normal termination

****NOTE**** Abnormal termination is used to go back to a point on the flight path and start over.

****NOTE**** The significance of these four (4) keys only applies to aircraft milestone entry. Any numbered key is appropriate for entering gun/missile locations.

ARE YOU FINISHED WITH THIS FLIGHT PATH: 0=NO 1=YES

Answer with 1, the 0 isn't working at present.

OUTPUT FILE: 0=NO OUTPUT; 1=OUTPUT

In order to submit your data to either P001 or MICE you must have generated both a guns/missile AND a flight path file. Until both are done or when experimenting without a desire for survival statistics, select 0=NO OUTPUT and IBMPIP will terminate.

EXTENDED OUTPUT: 0=NOT WANTED: 1=EXTENDED OUTPUT

Select 0; there is a bug in selection 1 which does not affect the operation of the game.

MISSILE TYPE BETWEEN 1 AND 7

Select 5 only; the performance data on the other missiles are not contained in this instructional package.

ENTER YOUR USER ID NUMBER: -- WITHOUT THE "p"

The questions you are now answering are for the automatic generation of JCL (Job Control Language) cards that will be inserted into the P001 DATA A1 and/or the MICE DATA A1 files for submission to batch processing.

At this time refer to Appendix B for procedures on entering gun emplacements.

** CHOICE 1 NEGATES GOING TO ALTIMETER

The advantage of choice 1 to the user is the convenience of not going over to the altitude bar for the guns/missile; each time a battery is emplaced, the map border will flash again, prompting the input of the next battery. Should 2 be selected, there is no program limit on how high a gun may be emplaced. Consult with the instructor to determine his wishes.

ERROR CHECKING: 0=NO CHECKING, 1=CHECKING

You have all ready become acquainted with the possible gun/missile errors. It is recommended that error checking be not selected, for the following reason. As earlier mentioned, error checking will trigger on flight path data being read in from a data file. At present, once guns/missile are entered, the user must still either generate a flight path at the terminal, or read one in from a data file in order for program to progress normally.

ENTER GUN LOCATIONS

Look at the bottom right hand corner of the IBM screen. You will see MORE... You are being prompted to hit the CLEAR key. At this time, the map border will flash and the cross hairs will appear again.

At this time look sequentially at Figs. 9,10,11,12,13,14,&15. The fixed, type 5 battery is all ready in place. In each succeeding frame, another battery is emplaced.

Now note the inner bold ring. This is the 3000 mtr limit, outside of which your type 3 weapons must be emplaced. Note Figs. 14 and 15.

When the last user definable battery is emplaced, the IBM screen will illuminate with the following message: ENTER MISSILE LOCATION along with the now familiar MORE... Again hit the CLEAR key and enter the missile location. Observe Fig. 16 carefully, as the missile symbol is not large. (X=8,500,Y=9,500).

At this point you will be prompted with ENTER AIRCRAFT MILESTONES. Proceed as discussed under Aircraft.

E. AT COMPLETION OF SESSION

When you are all done, and IBMPIP is terminated you should have a GUN LOC file and a PTS LOC file residing on your A1 disk. If you are not ready at this time to conduct file exchanges with the opposing team you release the 222 B disk with the command:

REL 222

then log off or whatever.

F. HOW TO MAKE A HARD COPY

To make a hard copy of your map, reach over and hit the HARD COPY button on the graphics screen. A loud racket will indicate which printer is making the copy. Most of the graphics screens indicate to which printer they are attached. You may have to turn on the printer as they are noisy and are frequently left off until needed. If you turn the printer on, give it 30 seconds to 1 minute to warm up or copy quality will be non-existent or severely degraded. Darkness of the copy is controlled by the HARD COPY INTENSITY knob under the graphics screen. CLOCKWISE increases darkness.

G. PRIOR TO EXCHANGING FILES

Exchanging files is the method of attacking or defending against the opposing team. If a file is sent to another user and that user has a file by the same name the incoming file will overwrite the existing file destroying the existing file unless precautions are taken.

The following is recommended; RED team; rename your pts loc file PTSR LOC BLUE team rename your pts loc file PTSB LOC

This can be done inside FLIST in the following manner; Place cursor beside PTS LOC and type

RENAME / PTSR = =

and hit enter. (The BLUE team, of course, would use PTSB) Now your file is protected.

(BLUE/RED) team; SEND your (PTSB/PTSR) LOC file to the opposition. If the respective teams do not have a SEND EXEC file, one may be obtained from the Aero Disk. Link up to the disk using the previous instructions. When linked, issue the following command,

SEND ,(you user number) .SEND EXEC.

Example; SEND 2408p SEND EXEC

Don't forget to release 222 when done. For instructions on how to use SEND, merely type SEND. It will also tell the receiver how to load the PTSx LOC file sent to them. To load this SEND EXEC on your disk issue either of the following commands;

RDR

DISK LOAD

Once the files are received, rename them back to PTS LOC. Link to the Aero Disk using the described procedures. Run IEMPIP and select

0=DISK for guns 0=DISK for aircraft

with ERROR CHECKING off. At the completion of your run the P001 and MICE printouts will report on how well the oppositions aircraft did against your defense. The other team will (hopefully not gleefully) report to you how well your flight fared against their defense. Good Luck.

APPENDIX B
DERIVATION OF EQUATIONS USED IN TEXT

From [Ref. 3], instantaneous velocity is given as;

$$v = \frac{ds}{dt}$$

and instantaneous acceleration is given as:

$$a = \frac{dv}{dt}$$

If the first equation is solved for dt and substituted into the second equation, the following results:

$$ads = vdv$$

If the acceleration is presumed constant over finite distances, the above equation may be integrated:

$$a \int_0^s ds = \int_{v_{old}}^{v_{new}} v dv$$

$$a = \frac{v_{new}^2 - v_{old}^2}{2s}$$

$$ACC = (VELON^2 - VELOO^2) / (2 * AJ1)$$

Returning to [Ref. 3], for;

$$a = \frac{dv}{dt}$$

if the equation is recast as;

$$dv = a dt$$

and the presumption of constant acceleration is made, the following integration results;

$$\int_{V_{OLD}}^{V_{NEW}} dv = a \int_0^t dt$$

$$V_{NEW} - V_{OLD} = at$$

$$V_{NEW} = V_{OLD} + at$$

Substitute this equation into;

$$v = \frac{ds}{dt}$$

and integrate to get;

$$ds = v dt$$

$$\int_0^s ds = \int_0^t (v_{old} + at) dt$$

$$s = v_{old} t + \frac{1}{2} at^2$$

$$DELX = UOLD * DELTEE + .5 * UDOT * DELTEE ** 2$$

Reaching back;

$$\int_{v_{old}}^{v_{TEMP}} dv = a \int_0^t dt$$

$$v_{TEMP} = v_{old} + at$$

$$UTEMP = UOLD + UDOT * DELTEE$$

From [Ref. 4], for a steady coordinated turn;

$$\text{TURNRATE (deg/sec)} = \frac{1091 \tan \phi (d)}{V}$$

(ϕ) = bank angle (degrees)

V = velocity (knots) TAS

Convert the turn rate to radians/sec:

$$\text{TURNRATE (r/s)} = \frac{1091 \tan \phi (r)}{V} \cdot \frac{2\pi}{360}$$

Convert V(knots) to V(meters/sec):

$$\text{TURNRATE (r/s)} = \frac{1091 \tan \phi (r)}{(V \cdot 1.9438)} \cdot \frac{2\pi}{360}$$

Solve for angle of bank:

$$\phi (r) = \text{atan} \left(\frac{\text{TURNRATE (r/s)} \cdot V (m/s)}{9.7958} \right)$$

$$\text{PHINew} = \text{ATAN}(\text{TURNRATE} \times \text{VELOCITY} / 9.7958)$$

From [Ref. 5], the pressure ratio as a function of temperature and altitude, for the model atmosphere is:

$$\int_{p_0}^p dp = -\frac{G}{R} \int_0^H \frac{dH}{T_0 + (dT/dH)H}$$

$$\frac{p}{p_0} = \left[1 + \left(\frac{dT}{dH} \right) \left(\frac{H}{T_0} \right) \right]^{\left(\frac{G/R}{-dT/dH} \right)}$$

where

$p(\text{naught})$ = standard sea-level atmospheric pressure

dT/dH = temperature lapse rate in the troposphere

H = geopotential altitude (assumed equal to geometric or tapeline altitude for low altitudes)

$T(\text{naught})$ = standard sea-level atmospheric temperature

G = acceleration due to gravity (assumed to be constant)

R = universal gas constant

The temperature profile for the same model atmosphere is:

$$T = T_0 + (dT/dH)H$$

Applying the perfect gas law to get density profile:

$$\frac{\rho}{\rho_0} = \frac{\left[1 + \left(\frac{dT}{dH} \right) \left(\frac{H}{T_0} \right) \right]^{\left(\frac{G/R}{-dT/dH} \right)}}{\left[1 + \left(\frac{dT}{dH} \right) \left(\frac{H}{T_0} \right) \right]} \\ = \left[1 + \left(\frac{dT}{dH} \right) \left(\frac{H}{T_0} \right) \right]^{\left(\frac{G/R}{-dT/dH} - 1 \right)}$$

The temperature lapse rate in SI units is:

$$\frac{dT}{dH} = -6.5 \times 10^{-3} \text{ } ^\circ\text{K/m}$$

Gravitational acceleration in SI units is:

$$G = 9.807 \text{ m/s}^2$$

The universal gas constant is:

$$287 \frac{\text{N}\cdot\text{m}}{\text{kg}\cdot^\circ\text{K}}$$

Thus;

$$\frac{G/R}{-dT/dH} - 1 = 4.257$$

and sea-level standard air density is:

$$\rho_0 = 1.225 \frac{\text{N}\cdot\text{s}^2}{\text{m}^4}$$

APPENDIX C
FLIGHT DYNAMICS AND COORDINATE TRANSFORMATIONS

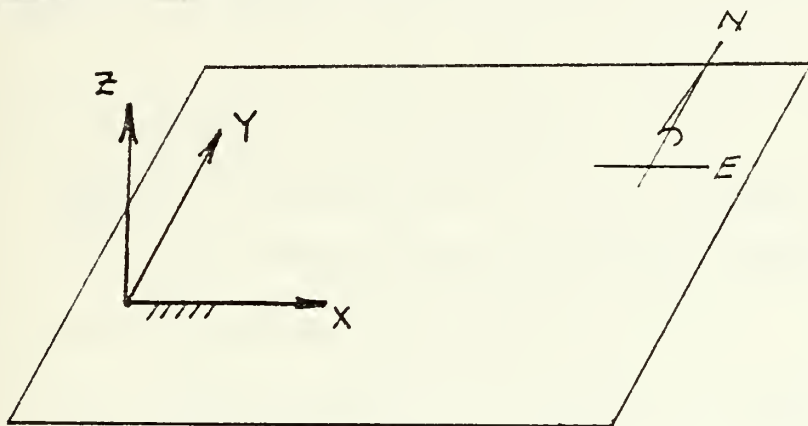


Figure C.1 The Map Coordinate System.

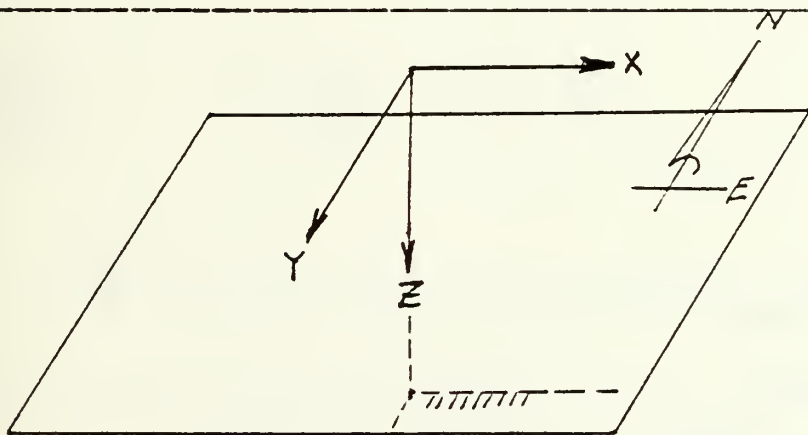


Figure C.2 The Aerodynamicist's Coordinate System.

The following development is taken from [Ref. 6]. For the equation (linear momentum = applied forces) for an aircraft with a constant mass (fuel burn not accounted for);

$$m \frac{d\vec{V}_e}{dt} = mg + F_a + F_t$$

m = mass of the airplane

$\left(\frac{d\vec{V}_e}{dt}\right)$ = the acceleration of the aircraft in the earth-fixed coordinate system

mg = force applied due to gravity

F_a = force applied by aerodynamic loading

F_t = force applied by thrust

This equation is transformed from the earth-fixed coordinate system to a coordinate system attached to the aircraft;

$$m \frac{d\vec{V}_a}{dt} = m \left(\frac{d\vec{V}_a}{dt} + \omega \times \vec{V}_a \right)$$

$\left(\frac{d\vec{V}_a}{dt}\right)$ = acceleration of the aircraft in the aircraft-fixed coordinate system

(ω) = rotational velocity of the aircraft-fixed coordinate system with respect to the earth-fixed coordinate system

If the above equation is expanded into its orthogonal components:

$$m\dot{u}_e = m (\dot{u}_a - v_a R + w_a Q) = mg_x + F_{a_x} + F_{t_x}$$

$$m\dot{v}_e = m (\dot{v}_a + u_a R - w_a P) = mg_y + F_{a_y} + F_{t_y}$$

$$m\dot{w}_e = m (\dot{w}_a - u_a Q + v_a P) = mg_z + F_{a_z} + F_{t_z}$$

The above is termed the aircraft equations of motion. In the aircraft-fixed coordinate system, the components are shown in figure C.3. The next three illustrations show the transformation of earth-fixed linear velocities and accelerations to aircraft-fixed linear velocities and accelerations: Finite rotational displacements do not act as vectors, hence the requirement for the order of the above rotations. This is demonstrated in figure C.7. Transformation of earth-fixed rotational velocities into aircraft-fixed velocities involves the following mapping.

$$\vec{\omega} = \dot{\psi} \vec{e}_1 + \dot{\theta} \vec{e}_2 + \dot{\phi} \vec{e}_3 \quad \Rightarrow \quad \vec{P} + \vec{Q} + \vec{R}$$

In terms of unit vectors, (ω) earth-fixed is presented;

$$\vec{\omega}_e = \dot{\psi} \hat{k}_2 + \dot{\theta} \hat{j}_3 + \dot{\phi} \hat{i}_a$$

looking at figure C.8, it is seen that;

$$\hat{k}_2 = -\sin\theta \hat{i}_3 + \cos\theta \hat{k}_3 = -\sin\theta \hat{i}_a + \cos\theta \hat{k}_3$$

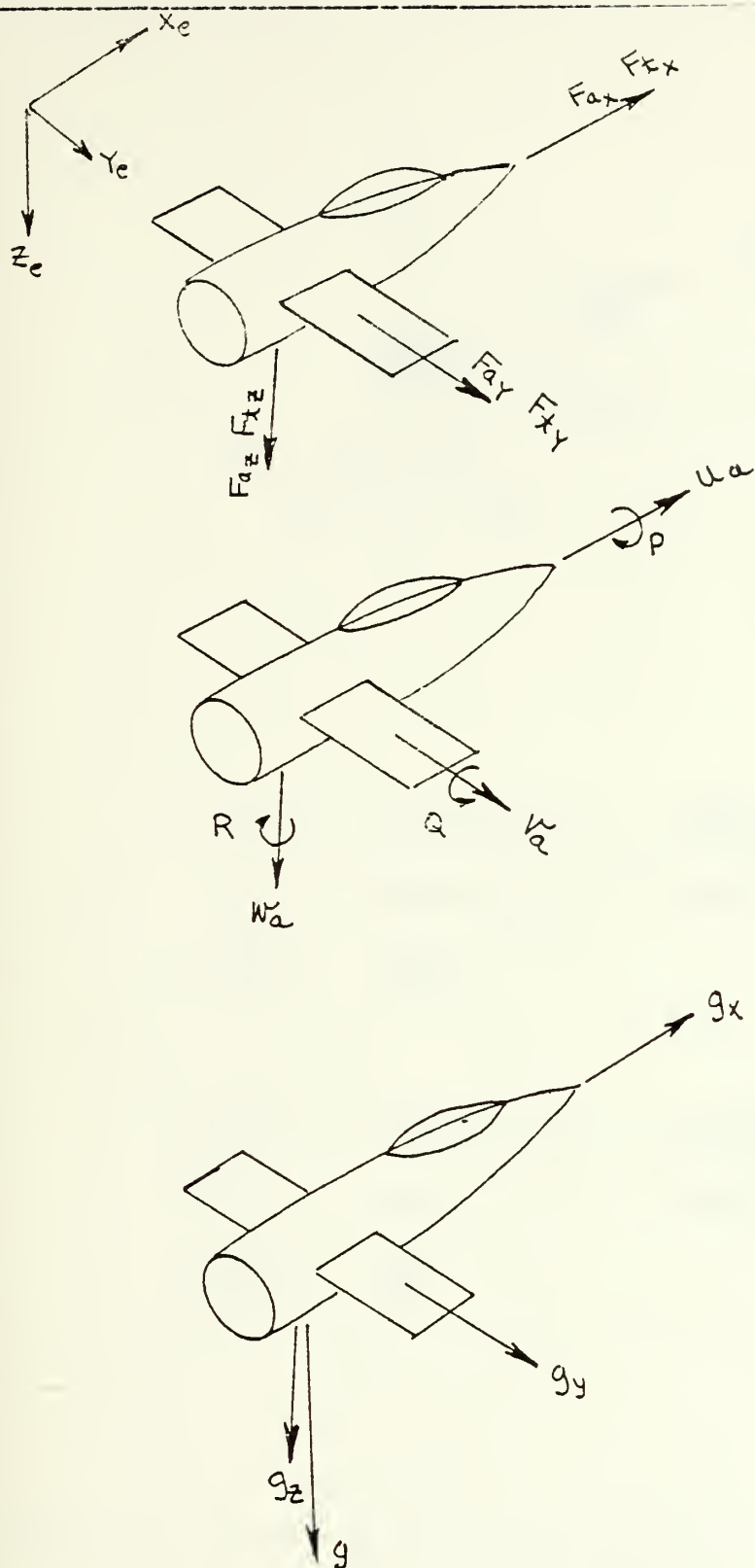
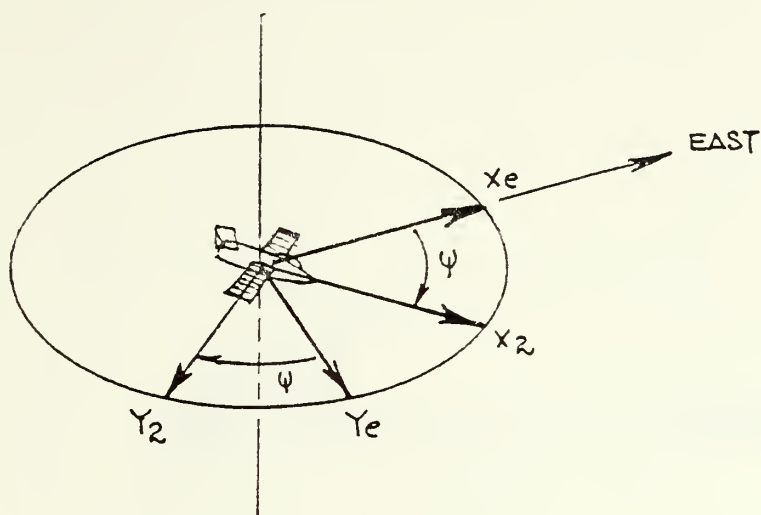


Figure C.3 Components of the Equations of Motion.



$$u_2 = \cos \psi u_e + \sin \psi v_e$$

$$v_2 = -\sin \psi u_e + \cos \psi v_e$$

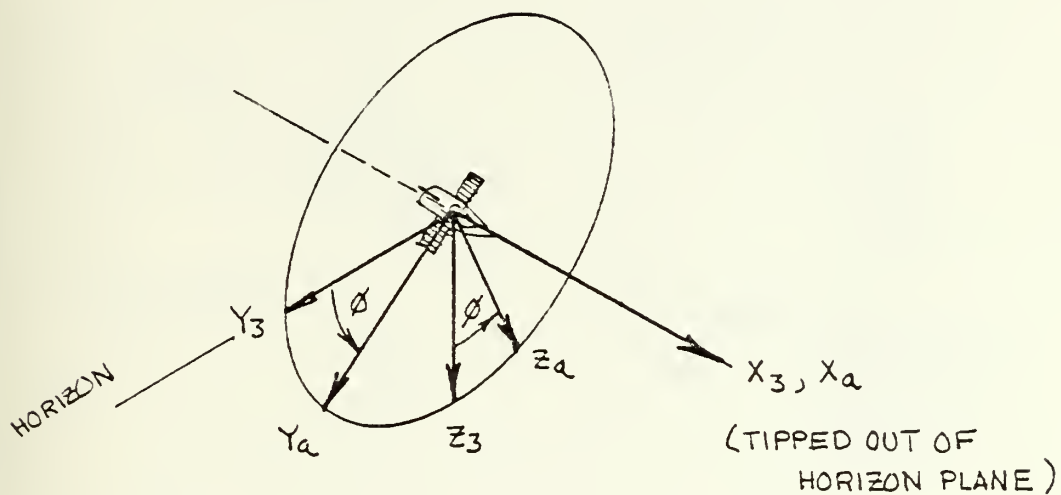
$$w_2 = w_e$$

$$\dot{u}_2 = \cos \psi \dot{u}_e + \sin \psi \dot{v}_e$$

$$\dot{v}_2 = -\sin \psi \dot{u}_e + \cos \psi \dot{v}_e$$

$$\dot{w}_2 = \dot{w}_e$$

Figure C.4 First Rotation.



$$\mu_a = \mu_3$$

$$v_a = \cos \phi \, v_3 + \sin \phi \, w_3$$

$$w_a = -\sin \phi \, v_3 + \cos \phi \, w_3$$

$$\dot{\mu}_a = \dot{\mu}_3$$

$$\dot{v}_a = \cos \phi \, \dot{v}_3 + \sin \phi \, \dot{w}_3$$

$$\dot{w}_a = -\sin \phi \, \dot{v}_3 + \cos \phi \, \dot{w}_3$$

Figure C.6 Third Rotation.

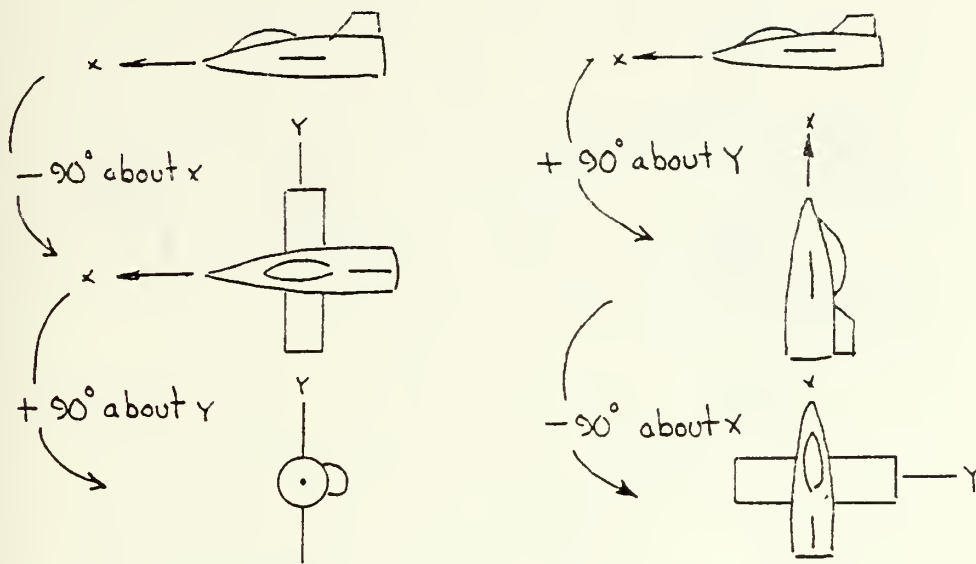


Figure C.7 Finite Rotations Do Not Act As Vectors.

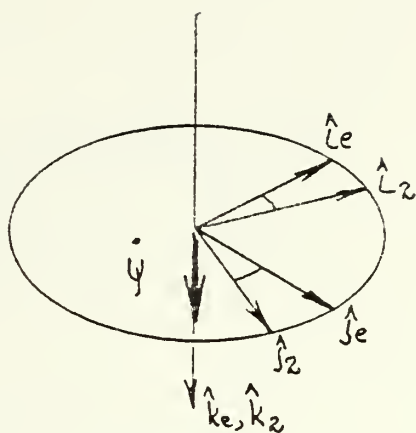
furthermore; looking at C.8,

$$\hat{k}_3 = \sin \phi \hat{j}_a + \cos \phi \hat{k}_a$$

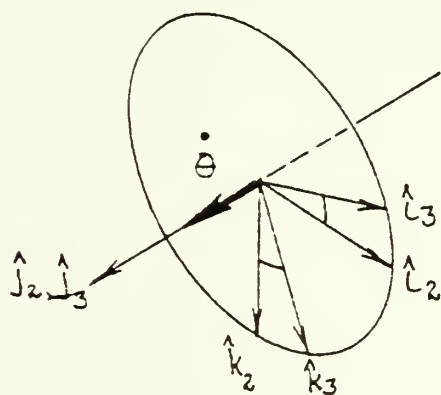
and finally, looking at the same figure,

$$\hat{j}_3 = \cos \phi \hat{j}_a - \sin \phi \hat{k}_a$$

$$\vec{\dot{\psi}} = \dot{\psi} \hat{k}_e = \dot{\psi} \hat{k}_2$$



$$\vec{\dot{\theta}} = \dot{\theta} \hat{j}_2 = \dot{\theta} \hat{j}_3$$



$$\vec{\dot{\phi}} = \dot{\phi} \hat{l}_3 = \dot{\phi} \hat{l}_a$$

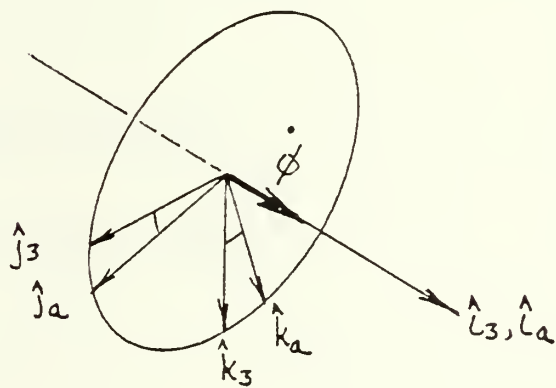


Figure C.8 Transformation of $(\dot{\omega})$.

Substituting into (6) in terms of the aircraft direction vectors;

$$\vec{\omega} = -\sin\theta \hat{l}_a + \cos\theta (\sin\phi \hat{j}_a + \cos\phi \hat{k}_a) \dot{\psi} + (\cos\phi \hat{j}_a - \sin\phi \hat{k}_a) \dot{\theta} + \dot{\phi} \hat{l}_a$$

and this must equal

$$\vec{\omega} = P \hat{l}_a + Q \hat{j}_a + R \hat{k}_a$$

Therefore, solving for components,

$$P = \dot{\phi} - \sin(\theta) \dot{\psi}$$

$$Q = \cos(\phi) \dot{\theta} + \cos(\theta) \sin(\phi) \dot{\psi}$$

$$R = -\sin(\phi) \dot{\theta} + \cos(\theta) \cos(\phi) \dot{\psi}$$

APPENDIX D
CONSOLIDATED COMMON LISTING

```
COMMON/ABC1/WX(1000), WY(1000), WZ(1000),
*      WU(1000), WV(1000), WW(1000),
*      WH(1000), WP(1000), WR(1000),
*      WG(1000), T(1000), ICT
```

```
COMMON/ABC2/XNEW, DELX, XTEMP, XOLD,
*      YNEW, DELY, YTEMP, YOLD,
*      ZNEW, DELZ, ZTEMP, ZOLD
```

```
COMMON/ABC3/VELCN, VELOT, VELOO, N,
*      UNEW, UTEMP, UOLD,
*      VNEW, VTEMP, VOLD,
*      WNEW, WTEMP, WOLD
```

```
COMMON/ABC4/AG1, AH1, AI1, AJ1, AJ2
```

```
COMMON/ABD1/AA1, AB1, AC1
```

```
COMMON/ABD2/VSTALL, VMAX, VMAX1, VMAX2, DELTEE, DELTV, DELTZ
```

```
COMMON/ABD3/PSIDOT, THEDOT, PHIDOT
```

```
COMMON/ABD4/PSINEW, THENEW, PHINEW,
*      PSIOLD, THEOLD, PHIOLD
```

```
COMMON/ABF1/U1, V1, W1, U2, V2, W2, U3, V3, W3
```

```
COMMON/ABF2/UD1, VD1, WD1, UD2, VD2, WD2, UD3, VD3, WD3
```

```
COMMON/ABF3/UAC, VAC, WAC, UDOTAC, VDOTAC, WDOTAC, P, Q, R
```

```
COMMON/LOC/X1, Y1, Z1, V1, LTR, LTR2
```

```
COMMON/MN/ IBAUD
```

```
COMMON/MN1/MINX, MAXX, MINY, MAXY, AINX1, MAXX1, MINX2, MAXX2
```

```
COMMON/NAM/INAM1(2), INAM2(2), IUNR, IANR, IDEST
```

```
COMMON/OPT/IGUN, IPNCH, IEXT, ISAM, IMP, KER, IMPZ
```

```
COMMON/PAR2/XGUN(7), YGUN(7), ZGUN(7), XSAM, YSAM, ZSAM, GR(7)
```


*COMMON/PAR4/APPEMAX,HTMIN,HTMAX,POPMIN,BAALT,BRMIN,BRMAX,
BRMIN,BRRMAX,RADTODEG,HTRTOFT

COMMON/TAR/TARGX,TARGY

APPENDIX E

EXECS USED TO DRIVE FORTRAN PROGRAMS

```

C*****
C IBMPIP1 EXEC A1
C THE FOLLOWING EXEC DRIVES IBMPIP1 -- D.A.M. 17MAR84 --
C*****
STRACE ERR
EXEC GRAF77
FILEDEF 09 DISK PICTUR DATA B1 (RECFM F LRECL 80
FILEDEF 10 DISK GUN LCC A1 (RECFM F LRECL 80
FILEDEF 11 DISK PTS LCC A1 (RECFM F LRECL 80
FILEDEF 16 DISK PAR SAV A1 (RECFM F LRECL 80
FILEDEF 18 DISK P001 DATA A1 (RECFM F LRECL 80
FILEDEF 17 DISK MICE DATA A1 (RECFM F LRECL 80
FILEDEF 19 DISK TEMP DATA A1 (RECFM F LRECL 80
LOAD IBMPIP1 (START

```

```

C*****
C IBMPIP2 EXEC A1
C THE FOLLOWING EXEC WAS USED TO DRIVE IBMPIP2 -- D.A.M. 17MAR84 --
C*****
STRACE ERR
EXEC GRAF77
FILEDEF 09 DISK PICTUR DATA A1 (RECFM F LRECL 80
FILEDEF 10 DISK GUN LCC A1 (RECFM F LRECL 80
FILEDEF 11 DISK PTS LCC A1 (RECFM F LRECL 130 BLOCK 130
FILEDEF 16 DISK PAR SAV A1 (RECFM F LRECL 80
FILEDEF 18 DISK P001 DATA A1 (RECFM F LRECL 80
FILEDEF 17 DISK MICE DATA A1 (RECFM F LRECL 80
LOAD IBMPIP2 (START

```

```

C*****
C GRAF77 EXEC A1
C THE FOLLOWING EXEC LINKS THE USER TO THE GRAPHICS IMPLEMENTATION
C -- D.A.M. 17MAR84 --
C*****
STRACE ERR
CP LINK GRAF77 191 199 RR
ACC 199 G
GLOBAL TITLIB MOD2EEH PORTMOD2 GRAFLIB CMSLIB NONIMSL

```


CLEANED UP JOHNS VERSION IBMPIP

115


```

C DRAW SAM SITE
C -----
      CALL GUNLCC (XSAM,YSAM,150.)
      WRITE (10,350) XSAM,YSAM,ZSAM
      GO TO 13C
C -----
C IGUN=2==> READ GUN SITES FROM DATA FILE
C -----
90    IF (IGUN.NE.0) GO TO 110
      DC 100 I=1,6
100    READ (10,350) XGUN(I),YGUN(I),ZGUN(I)
      READ (10,350) XSAM,YSAM,ZSAM
C -----
C DRAW GUN SITES AND ENGAGEMENT CIRCLES
C -----
110   DC 120 I=1,6
120   CALL GUNLCC (XGUN(I),YGUN(I),GE(I))
      CALL GUNLCC (XSAM,YSAM,150.)
130   CONTINUE
C -----
C THIS SECTION ACCEPTS THE FLIGHT MILESTONES
C -----
      PTFAC=3.28084
      DGFAC=57.29578
      CALL EBRMK (15)
      REWIND 11
140   MNUM=MNUM+1
C -----
C READ MILESTONES FROM THE DISK
C -----
150   IF (IMP.NE.0) GO TO 160
      READ (11,360) X(MNUM),Y(MNUM),Z(MNUM),VEL(MNUM),LTR
      IF (X(MNUM).LE.0.1) GO TO 150
      IF (LTR.EQ.1) IMP=1
      LTR2=LTR
      GO TO 18C
C -----
C READ MILESTONES FROM THE TERMINAL
C -----
160   CALL XYPT
      X(MNUM)=X1
      Y(MNUM)=Y1
170   CALL ZVPT
      Z(MNUM)=Z1
      VEL(MNUM)=V1
      IF (IMP.EQ.0) GO TO 140
C -----
C COMPUTE FLIGHT PARAMETERS AND CHECK FOR ERRORS OR USER COMMANDS
C -----
180   IF (MNUM.EQ.1.AND.(LTR.EQ.83.CE.LTR2.EQ.83)) STOP
      IF (MNUM.EQ.1) GO TO 140
      IERR=0
      CALL VALSET (IERR)
      IF (((LTR.EQ.82).OR.(LTR2.EQ.82)).AND.(IMP.NE.0)) GO TO 200
      IF ((LTR.EQ.86).OR.(LTR2.EQ.86)) MBR=MNUM
      IF ((LTR.EQ.83).OR.(LTR2.EQ.83)) GO TO 230
      IF (IERR.EQ.0) CALL ERRCHK (IERR)
      IF ((IERR*NER.EQ.0).AND.(IERR.NE.12)) GO TO 190
      CALL EBRMK (IERR)
      WRITE (6,390)
      WRITE (6,400)
      READ (5,*) ICOR
      CALL PRTCMS ('CLRSCRN ')
      IF (ICOR.NE.1) GO TO 190
      WRITE (6,410)
      READ (5,*) IANS
      CALL PRTCMS ('CLRSCRN ')
      IF (IANS.NE.1) GO TO 160
      GO TO 17C
C -----
C MILESTONE ACCEPTED, RESTART THE SEQUENCE
C -----
190   CALL PTHFLT
      IF (LTR.EQ. 65 .OR. LTR2.EQ. 65) CALL AIMPT (MNUM)
      GO TO 140
C -----

```



```

C THIS SECTION RESETS THE DATA
C-----
200  CALL SCENE (1,ICT)
    REWIND 19
    IF (MBR.GT. ICT) MBR=0
C-----
C COMPLETE RESTART
C-----
    IF (ICT.GT. 1) GC TO 210
    MNUM=0
    GC TO 110
C-----
C DRAW RETAINED MILESTONES
C-----
210  IF ((ICT.GE. MNUM).AND.(MNUM.GE. 3)) ICT=MNUM-1
    DO 220 MNUM=2,ICT
220  CALL PTHFLT
    MNUM=MNUM-1
    GC TO 110
C-----
C THIS SECTION TERMINATES THE PROGRAM
C-----
230  BLANK=0.
    CALL PTHFLT
    LAST=IMP
    CALL ELFIN (LAST)
C-----
C COMPUTE FLIGHT PARAMETERS FOR FINAL LEG
C-----
    DX=X(MNUM)-X(MNUM-1)
    DY=Y(MNUM)-Y(MNUM-1)
    DZ=Z(MNUM)-Z(MNUM-1)
    RA(MNUM)=0.
    CA(MNUM)=ATAN2(DZ,SQRT(DX**2 + DY**2))
    IF ((DX.EQ. 0.).OR.(DY.EQ. 0.)) GO TO 240
    HDG(MNUM)=ATAN2(DY,DX)
    GC TO 250
240  IF ((DX.EQ. 0.) .AND. (DY.GT. 0.)) HDG(MNUM)=1.57079
    IF ((DX.EQ. 0.) .AND. (DY.LT. 0.)) HDG(MNUM)=-1.57079
    IF ((DY.EQ. 0.) .AND. (DX.LT. 0.)) HDG(MNUM)=3.14159
    IF ((DY.EQ. 0.) .AND. (DX.GT. 0.)) HDG(MNUM)=0.
250  XDCT(MNUM)=VEL(MNUM)*COS(HDG(MNUM))*COS(CA(MNUM))
    YDCT(MNUM)=VEL(MNUM)*SIN(HDG(MNUM))*COS(CA(MNUM))
    ZDCT(MNUM)=VEL(MNUM)*SIN(CA(MNUM))
C-----
C CREATES NEW "PTS LOC" IF NEW WEAPON OR MILESTONE COORDINATES
C HAVE BEEN INPUT
C-----
    IF ((IMP.EQ. 0).AND.(IGUN.NE. 1)) GO TO 280
    REWIND 11
    IGUN=1
    WRITE (11,360) BLANK,BLANK,BLANK,BLANK,BLANK
    DO 260 I=1,MNUM
    LTR=0
    IF (I.EQ. MBR) LTR=66
    IF (I.EQ. MNUM) LTR=LAST
260  WRITE (11,360) X(I),Y(I),Z(I),VEL(I),LTR
    WRITE (11,370)
    WRITE (11,380) VEL(1),MBR,BLANK,BLANK,IGUN,IGUN,BLANK,BLANK,IGUN,B
    LANK
    IF (IGUN.EQ. 0) GO TO 280
    DO 270 I=1,M
270  WRITE (11,360) XGUN(I),YGUN(I),ZGUN(I)
    WRITE (11,360) XSAM,YSAM,ZSAM
C-----
C CONVERT ANGLES FROM RADIANS TO DEGREES
C-----
280  DO 290 I=1,MNUM
    CA(I)=CA(I)*DGFAC
    RA(I)=RA(I)*DGFAC
290  HDG(I)=HDG(I)*DGFAC
C-----
C WRITE THE JCL CARDS
C-----
    IF (IPNCH.LT.1) STOP
    IF (IPNCH.EQ.2) GC TO 300

```



```

WRITE (6,420)
WRITE (6,430)
WRITE (6,440)
WRITE (6,450)
300 READ (5,530) INAM1(1),INAM1(2)
CALL FRTCMS ('CLRSCN ')
IF (IPNCH.EQ.1) GO TO 310
WRITE (6,460)
WRITE (6,430)
WRITE (6,440)
WRITE (6,450)
310 READ (5,530) INAM2(1),INAM2(2)
CALL FRTCMS ('CLRSCN ')
WRITE (6,470)
READ (5,*) IUNE
CALL FRTCMS ('CLRSCN ')
WRITE (6,480)
READ (5,*) IANR
CALL FRTCMS ('CLRSCN ')
C-----
C CHOOSE DESTINATION FOR OUTPUT: PRINTER OR READER FILE
C-----
320 WRITE (6,490)
WRITE (6,500)
READ (5,*) IDEST
IF ((IDEST.NE.1).AND.(IDEST.NE.2)) GO TO 320
CALL FRTCMS ('CLRSCN ')
C-----
C PRESET LOADS MICE/P001 DATA FILES ONTO USERS A1 DISK
C-----
CALL PRESET
C-----
C DUMMY INPUT ROUTINE TO LET USER KNOW WHERE S/HE IS IN THE PGM
C-----
WRITE (6,510)
READ (5,*) IGC
CALL FRTCMS ('CLRSCN ')
C-----
C FORCE CALL TO CMS TO SUBMIT DATA FILES TO BATCH PROCESSING PGM
C-----
IF (IPNCH.EQ.2) GO TO 330
CALL FRTCMS ('EXEC ', 'P001 ', 'DATA ')
IF (IPNCH.EQ.1) GO TO 340
330 CALL FRTCMS ('EXEC ', 'MICE ', 'DATA ')
340 CONTINUE
C-----
C DUMMY INPUT ROUTINE TO LET USER KNOW PGM TERMINATION HAS BEEN REACHED
C-----
WRITE (6,520)
READ (5,*) IGO
STCP
350 FORMAT (3F10.2)
360 FCRMAT (4F10.0,I3)
370 FORMAT (6H99999.)
380 FCRMAT (F10.0,I2,8I1,F10.0)
390 FORMAT (3H )
400 FCRMAT (54H DO YOU WANT TO FIX THE ERROR:0=NO,USE THE POINT;1=YES)
410 FORMAT (4CH DO YOU WANT TO RETAIN X-Y PT:1=YES;2=NO)
420 FCRMAT (40H TO SUBMIT P001 TO BATCH YOU MUST CREATE)
430 FORMAT (31H A JCB CONTROL LANGUAGE "CARC".)
440 FORMAT (51H ENTER A NAME -- IT MAY CONTAIN UP TO 8 CHARACTERS;)
450 FCRMAT (25H **DO NOT INSERT BLANKS**)
460 FORMAT (44H TO SUBMIT **MICE** TO BATCH YOU MUST CREATE)
470 FCRMAT (46H ENTER YOUR USER ID NUMBER -- WITHOUT THE "P".)
480 FCRMAT (48H ENTER YOUR ACCT NUMBER -- PROVIDED BY THE PROF.)
490 FORMAT (41H CHOOSE THE DESTINATION FOR YOUR OUTPUT: )
500 FCRMAT (30H 1=PRINTER; 2=YOUR READER FILE)
510 FORMAT (46H ENTER ANY **NUMBER** TO SUBMIT YOUR P001/MICE,14H DATA
1 FILE(S).)
520 FORMAT (40H ENTER ANY **NUMBER** TO TERMINATE T2PIP)
530 FORMAT (2A4)
END

```



```

C*****
C SUBROUTINE BEGIN      THIS SUBROUTINE INITIALIZES THE GRAPHICS AND REQUESTS
C                        THE USER OPTIONS.
C*****
      SUBROUTINE BEGIN
      COMMON /MN/ IEAUC
      COMMON /OPT/ IGUN, IENCH, IEXT, ISAM, IMP, KER, IMPE
C-----
C READ USER OPTIONS
C-----
10    CALL FRTCMS ('CLRSCLR ')
      WRITE (6,60)
      READ (5,*) IGUN
      IF ((IGUN.NE.0).AND.(IGUN.NE.1).AND.(IGUN.NE.2)) GO TO 10
      CALL FRTCMS ('CLRSCLR ')
      IF (IGUN.NE.1) GO TO 20
      WRITE (6,70)
      WRITE (6,80)
      READ (5,*) IMPZ
20    CALL FRTCMS ('CLRSCLR ')
      WRITE (6,90)
      READ (5,*) IMP
      IF ((IMP.NE.0).AND.(IMP.NE.1)) GO TO 20
30    CALL FRTCMS ('CLRSCLR ')
      WRITE (6,100)
      READ (5,*) KER
      IF ((KER.NE.0).AND.(KER.NE.1)) GO TO 30
40    CALL FRTCMS ('CLRSCLR ')
      WRITE (6,110)
      WRITE (6,120)
      READ (5,*) IEAUC
      IF ((IEAUC.NE.0).AND.(IEAUC.NE.1)) GO TO 40
50    CALL FRTCMS ('CLRSCLR ')
      WRITE (6,130)
      READ (5,*) KON
      IF ((KON.NE.1).AND.(KON.NE.2)) GO TO 50
      CALL FRTCMS ('CLRSCLR ')
      IF (KON.NE.1) CALL CONCEG
C-----
C INITIALIZE THE GRAPHICS AND CLEAR SCREEN
C-----
      CALL DSINIT
      CALL GSERSE
      RETURN
60    FORMAT (56H GUNS:0=DISK FILE;1=TERMINAL;2=PRESET(GOUGE SET OF GUNS
1) )
70    FORMAT (55H ** CHOICE 1 NEGATES HAVING TO GO OVER TO THE ALTIMETER
1) )
80    FORMAT (55H GUN/SAM SITE ALTITUDE:1=FIXED TO 20M;2=USER DEFINEABLE
1) )
90    FORMAT (50H AIRCRAFT MILESTONE INPUT: 0=DISK FILE; 1=TERMINAL)
100   FORMAT (50H ERROR CHECKING: 0=NO CHECKING; 1=CHECK FOR ERRORS)
110   FORMAT (54H ** SELECT 1200 BAUD FOR THE IEM3277/TEK618 TERMINALS.)
120   FORMAT (35H BAUD RATE: 0=300 BAUD; 1=1200 BAUD)
130   FORMAT (50H FLIGHT & GAME PARAMETERS: 1=DEFAULT; 2=USER INPUT)
      END

```



```

C*****
C SUBROUTINE CONCHG THIS SUBROUTINE ALLWS OPERATOR CHANGES TO THE
C                      SIMULATION PARAMETERS.
C*****
      SUBROUTINE CONCHG
      COMMON /PAR3/ TMD,ACLIFT,CLMAX,WL,TMAX,CDO,CDK,VMAX1,VMAX2
      COMMON /PAR4/ APPMAX,HTMIN,HTMAX,SPDMIN,GMAX,POPMIN
      COMMON /TAR/ TARGX,TARGY
C-----
C WRITE INSTRUCTIONS
C-----
10  WRITE (6,190)
    WRITE (6,200)
    WRITE (6,210)
    WRITE (6,220)
    WRITE (6,230)
    WRITE (6,240)
    WRITE (6,250)
    WRITE (6,260)
    WRITE (6,270)
    READ (5,*) K
    IF ((K.LE.0).OR.(K.GT.17)) GO TO 10
    GO TO (180,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170
1) , K
20  WRITE (6,280)
    READ (5,*) CLMAX
    GO TO 10
30  WRITE (6,290)
    READ (5,*) WL
    GO TO 10
40  WRITE (6,300)
    READ (5,*) SPDMIN
    GO TO 10
50  WRITE (6,310)
    READ (5,*) GMAX
    GO TO 10
60  WRITE (6,320)
    READ (5,*) HTMIN
    GO TO 10
70  WRITE (6,330)
    READ (5,*) HTMAX
    GO TO 10
80  WRITE (6,340)
    READ (5,*) POPMIN
    GO TO 10
90  WRITE (6,350)
    READ (5,*) APPMAX
    GO TO 10
100 WRITE (6,360)
    READ (5,*) TMAX
    GO TO 10
110 WRITE (6,370)
    READ (5,*) CDO
    GO TO 10
120 WRITE (6,380)
    READ (5,*) CDK
    GO TO 10
130 WRITE (6,390)
    READ (5,*) VMAX1
    GO TO 10
140 WRITE (6,400)
    READ (5,*) VMAX2
    GO TO 10
150 WRITE (6,410)
    READ (5,*) TARGX,TARGY
    GO TO 10
C-----
C READ NEW PARAMETER VALUES FROM THE DISK
C-----
160 REWIND 16
    READ (16,450) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
    READ (16,450) POPMIN,APPMAX,TMAX,CDO,CDK,VMAX1
    READ (16,450) VMAX2,TARGX,TARGY
    GO TO 10
C-----

```


C WRITE PARAMETER VALUES TO THE TERMINAL

```

170  WRITE (6,420)
      WRITE (6,450) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      WRITE (6,430)
      WRITE (6,450) POPMIN,APPMAX,TMAX,CDO,CDK,VMAX1
      WRITE (6,440)
      WRITE (6,450) VMAX2,TARGX,TARGY
      GC TO 10

```

C SAVE PARAMETER VALUES ON DISK AND RETURN TO CALLING ROUTINE

```

180  REWIND 16
      WRITE (16,450) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      WRITE (16,450) POPMIN,APPMAX,TMAX,CDO,CDK,VMAX1
      WRITE (16,450) VMAX2,TARGX,TARGY
      RETURN
190  FORMAT (25H
200  FCORMAT (50H SELECT PARAMETER TO BE CHANGED: 1=NO MORE CHANGES)
210  FORMAT (48H 2=MAX LIFT COEFF; 3=WING LOADING; 4=STALL SPEED)
220  FCORMAT (47H 5=MAX G FORCE; 6=MIN ALT; 7=MAX ALT(<2200 MTR))
230  FCORMAT (42H 8=DISTANCE TO TARGET BEFORE POPUP ALLOWED)
240  FCORMAT (63H 9=MAX APPROACH ALT; 10=MAX T/W RATIO; 11=DRAG COEF. W/
10  LIFT)
250  FCORMAT (44H 12=LIFT DRAG CONSTANT; 13=MAX SPD WITH BOMB)
260  FCORMAT (44H 14=MAX SPD W/O BOMB; 15=TARGET COORDINATES)
270  FCORMAT (43H 16=READ PARAMETER FILE; 17=LIST PARAMETERS)
280  FCORMAT (27H ENTER MAX LIFT COEFFICIENT)
290  FCORMAT (19H ENTER WING LOADING)
300  FCORMAT (16H ENTER STALL SPEED)
310  FCORMAT (26H ENTER MAX G FORCE ALLOWED)
320  FCORMAT (23H ENTER MINIMUM ALTITUDE)
330  FCORMAT (47H ENTER MAXIMUM ALTITUDE (LESS THAN 2000 METERS))
340  FCORMAT (55H ENTER MINIMUM DISTANCE TO TARGET BEFORE POP-UP ALLOWED)
1)
350  FCORMAT (32H ENTER MAXIMUM APPROACH ALTITUDE)
360  FCORMAT (35H ENTER MAX THRUST TO WEIGHT ALLOWED)
370  FCORMAT (37H ENTER DRAG COEFFICIENT FOR ZERO LIFT)
380  FCORMAT (25H ENTER LIFT DRAG CONSTANT)
390  FCORMAT (32H ENTER MAX SPEED CARRYING A BOMB)
400  FCORMAT (39H ENTER MAX SPEED AFTER BOMB IS RELEASED)
410  FCORMAT (33H ENTER TARGET X AND Y COORDINATES)
420  FCORMAT (55H MAX CL WING LD STALL SPD MAX G MIN.HT. MAX.HT.
1)
430  FCORMAT (75H POP-UP DIST APPR.HT. MAX THRUST CDO DRAG CONSTANT
1) MAX SPD WITH BOMB)
440  FCORMAT (55H MAX SPD W/O BOMB TARGET X COORD. TARGET Y COORD.
1)
450  FCORMAT (6F12.4)
      ENC

```



```

C *****
C SUBROUTINE SCENE      THIS ROUTINE DRAWS THE ATTACK MAP AND ASSOCIATED
C                       DISPLAYS. DURING RESET IT OBTAINS THE LAST MILE-
C                       STONE THE USER WISHES TO RETAIN.
C *****
C SUBROUTINE SCENE (IRC,ICT)
C COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C COMMON /ERR/ MKERRA(3),MKERY(18)
C COMMON /EN/ IEAUC
C COMMON /PAR/ X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
1) COMMON /TAR/ TARGX,TARGY
C REAL*4 GX,GY
C LOGICAL *1CHARS(20)
C ICT=0
C -----
C ERASE THE TEK618 SCREEN
C -----
C CALL GSERSE
C CALL DSTERN
C -----
C READ USER OPTIONS: EORCER AND GUN/MISS POSITS OR FILL IN MAP ALSO.
C -----
C WRITE (6,100)
C READ (5,*) MAP
C IF (IRC.EQ. 0) GC TO 10
C WRITE (6,90)
C READ (5,*) ICT
C CALL PRICMS ('CL3SCREEN ')
C -----
C DRAW THE X,Y MAP
C THE ROUTINE READS X,Y PAIRS FROM FILE 09. A -2.0,0 INDICATES
C END OF FILE, -0.5,0 INDICATES A MOVE, ALL OTHER VALUES ARE PO-
C SITIVE AND RESULT IN A DRAW TO THAT POINT.
C -----
C RE-INITIALIZE THE TEK618 SCREEN
C -----
10 CALL DSINIT
C CALL WIN (MINX,MAXX,16000.,12000.,1)
C IF (MAP.EQ. 0) GC TO 40
C REWIND 9
20 READ (9,80) GX,GY
C READ (9,80) GX,GY
30 CALL GBMOVE (GX,GY)
C READ (9,80) GX,GY
C IF (GX.LT.-1) GC TO 40
C IF (GX.LT. 0) GO TO 20
C CALL GBDRAW (GX,GY)
C GO TO 30
C -----
C MARK THE TARGET: GUNLOC PUTS A + AT GUN/MISS POSIT AND CIRCUMSCRIBES
C -----
40 CALL GUNLOC (TARGX,TARGY,150.)
C CALL GUNLOC (TARGY,TARGY,300.)
C -----
C DRAW TIC MARKS EVERY 2000 MIRS
C -----
C DO 50 I=1,8
C IW=2*I
C -----
C THIS BLOCK DRAWS THE X-AXIS TIC MARKS
C -----
C CALL GBMCVE (I*2000.,1000.)
C CALL GBDRAW (I*2000.,0.)
C CALL GAXITC (1,IW,2,CHARS)
C CALL GBCHAR (I*2000.,0.,2.,0.,2,CHARS)
C -----
C THIS BLOCK DRAWS THE Y-AXIS TIC MARKS
C -----
C IF (I.GE. 6) GO TO 50
C CALL GBMCVE (1000.,I*2000.)
C CALL GBDRAW (0.,2000.*I)
C CALL GAXITC (1,IW,2,CHARS)
C CALL GBCHAR (-50.,I*2000.,2.,0.,2,CHARS)
50 CONTINUE

```



```

I1=MINY
I2=MAXY
MAXY=MINY-15
MINY=MINY-200
CALL WIN (MINX+275,MINX+2200,195.,185.,0)
CALL GBCHAR (10.,110.,2.5,0.,30,'KEY MARKS IN 2000 METER INTVLS')
CALL GBCHAR (10.,40.,2.5,0.,25,'ZERO DEGREES ALONG X AXIS')
CALL GSFRCE
MAXY=I2
MINY=I1
C-----
C DRAW THE ALTIMETER
C-----
CALL WIN (MINX1,MAXX1,350.,2200.,1)
DO 60 I=2,20,2
CALL GBMCVE (0.,I*100.)
CALL GBDRAW (350.,I*100.)
CALL GAXITC (1,I,2,CHARS)
60 CALL GBCHAR (0.,I*100.,2.,0.,2,CHARS)
DO 70 I=1,9
CALL GBMCVE (I*50.,100.)
70 CALL GBDRAW (I*50.,0.)
C-----
C MARK RESET POINT VELOCITY AND ALTIITUDE AS OPERATOR AID
C-----
IF (IBO.NE.0 .AND. ICT.GT.1) CALL GBGDOT (9.,VEL(ICT),Z(ICT))
MAXY=MINY-15
MINY=MINY-200
CALL WIN (MINX+2800,MINX+3900,195.,185.,0)
CALL GBCHAR (10.,110.,2.0,0.,25,'ALT MARKS IN 200 METER INTVLS')
CALL GBCHAR (10.,60.,2.0,0.,33,'VELO IN 50 M/S INTVL ALONG X AXIS')
1) CALL GSFRCE
MAXY=I2
MINY=I1
RETURN
80 FORMAT (2F10.2)
90 FORMAT (38H AT WHICH POINT DO YOU WANT TO RESTART)
100 FORMAT (38H VILLAGE: 0=NOT DRAWN; 1=VILLAGE DRAWN)
END

```



```

C *****
C SUBROUTINE WIN      THIS ROUTINE DEFINES A WINDOW EXTENDING FROM LX TO MX
C                     ON THE HORIZONTAL AXIS, AND FROM MINY TO MAXY ON THE
C                     VERTICAL AXIS. THE HORIZONTAL RANGE IS RX, THE
C                     VERTICAL RANGE IS RY.
C *****
C SUBROUTINE WIN (LX,MX,RX,RY,JMP)
C COMMON /MN/ IEAUX
C COMMON /MN/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C REAL*4 WIN1(4)
C INTEGER*4 V1(4)
C DATA WIN1(1)/C.,WIN1(2)/0./
C -----
C DEFINE THE WINDOW
C -----
C V1(2)=MINY
C V1(4)=MAXY
C WIN1(3)=RX
C WIN1(4)=RY
C V1(1)=LX
C V1(3)=MX
C CALL GASVIE (V1,0)
C CALL GASWIN (WIN1,0)
C IF (JMP .EQ. 0) GO TO 20
C -----
C FLASH THE DEFINED WINDOW
C -----
C ICT=4+50*IEAUX
C DO 10 I=1,ICT
C CALL GEMOVE (C.,0.)
C CALL GEDRAW (WIN1(1),WIN1(4))
C CALL GEDRAW (WIN1(3),WIN1(4))
C CALL GEDRAW (WIN1(3),WIN1(2))
C CALL GEDRAW (WIN1(1),WIN1(2))
10  RETURN
20  END

```



```

C *****
C SUBROUTINE GUNLOC
C      THIS SUBROUTINE DRAWS EFFECTIVE FIRE RADIUS CIRCLES
C      AROUND THE DEFENSIVE AAA BATTERIES. IT DRAWS
C      A RING WITHIN WHICH AIRCRAFT POP-UP MUST TAKE PLACE
C      AND A RING OUTSIDE WHICH THE TYPE 3 AAA BATTERIES
C      MUST BE PLACED.
C *****
C      SUBROUTINE GUNLOC (GX,GY,RAD)
C      COMMON /MN/ IBAUD
C      COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C      REAL*4 X,Y,DX,DY
C      X=GX
C      Y=GY
C -----
C IDENTIFY THE MAP AND PLACE A '+' AT THE LOCATION
C -----
C      CALL WIN (MINX,MAXX,18000.,12000.,0)
C      CALL GEMOVE (X-75.,Y)
C      CALL GBDRAW (X+75.,Y)
C      CALL GEMOVE (X,Y+75.)
C      CALL GBDRAW (X,Y-75.)
C -----
C DETERMINE NUMBER OF STEPS TO BE USED
C -----
C      ISTEP=3
C      IF (RAD.LT. 1600.) ISTEP=6
C -----
C DRAW A CIRCLE AROUND THE SITE
C -----
C      DO 10 I=3,180,ISTEP
C      ANGLE=I*.034907
C      DX=RAD*CCS(ANGLE)+X
C      DY=RAD*SIN(ANGLE)+Y
C      CALL GEMOVE (DX+10.,DY)
10    CALL GBDRAW (DX-10.,DY)
C -----
C DRAW 3KM & 6KM GUNSITE AND POP-UP LIMITATION RINGS
C -----
C      IF ((X.NE.14000.) .AND. (Y.NE.7220.)) RETURN
C      CALL GEGARC (3000.,3000.,X,Y,1.,1.,2.)
C      CALL GEGARC (6000.,6000.,X,Y,1.,1.,2.)
C      RETURN
C      END

```



```

C*****
C SUBROUTINE XYPT      THIS SUBROUTINE CONTAINS THE X AND Y COORDINATES
C                      AND COMMAND FOR EACH DESIGNATED POINT (MILESTONE).
C*****
C SUBROUTINE XYPT
C COMMON /LCC/ X1,Z1,Z1,V1,ITS,IIR2
C COMMON /EN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C COMMON /EN/ IEAUC
C CALL BELL
C-----
C FLASH MAP AND GET X,Y AND THE FIRST COMMAND
C-----
C CALL WIN (MINX,MAXX,18000.,12000.,1)
C CALL SPOT (X1,Y1,ITS)
C CALL BELL
C RETURN
C END

```

```

C*****
C SUBROUTINE ZVPT      THIS SUBROUTINE CONTAINS THE ALTITUDE AND, DEPENDING
C                      ON THE CIRCUMSTANCES, THE VELOCITY OF THE DESIGNATED
C                      POINT.
C*****
C SUBROUTINE ZVPT
C COMMON /LCC/ X1,Y1,Z1,V1,ITS,IIR2
C COMMON /EN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C COMMON /EN/ IEAUC
C-----
C FLASH ALTIMETER AND GET VEL,ALT AND THE SECOND COMMAND
C-----
C CALL WIN (MINX1,MAXX1,350.,2200.,1)
C CALL SPOT (V1,Z1,ITS2)
C RETURN
C END

```



```

C*****
C SUBROUTINE SFCOT      THIS RETURNS AN X-Y PAIR AND A COMMAND VALUE L1
C                       THE ROUTINE ALSO CREATES A DOT AT POINT X,Y.
C*****
      SUBROUTINE SFCOT (X,Y,L1)
      REAL*4 IX,TY,DCT
      INTEGER*4 L,1,CUT,KEY
C-----
C READ THE CURSOR AND PICK UP USER COMMAND
C-----
10    CALL GBRXYC (M,L,KEY,IX,TY)
      CALL GAXCTI (1,KEY,1,LEN,CUT)
      DCT=40.
C-----
C MARK THE SPOT
C-----
      CALL GBGDCI (DCT,IX,TY)
      X=IX
      Y=TY
C-----
C CONVERT THE COMMAND TO THE PROPER FORMAT
C-----
      L1=0
      IF (OUT.EQ.1) L1=65
      IF (OUT.EQ.2) L1=66
      IF (OUT.EQ.3) L1=82
      IF (OUT.EQ.4) L1=83
      RETURN
      END

```

```

C*****
C SUBROUTINE GUNCHK    THIS SUBROUTINE CHECKS GUN LOCATIONS AGAINST
C                       EMPLACEMENT RULES.
C*****
      SUBROUTINE GUNCHK (X,Y,IERR)
      COMMON /TAR/ TARGX,TARGY
C-----
C COMPUTE DISTANCE TO TARGET
C-----
      DIST=SQRT((X-TARGX)**2 + (Y-TARGY)**2)
      IF (DIST.LT. 3000.) IERR=13
      RETURN
      END

```

```

C*****
C SUBROUTINE SAMCHK    THIS SUBROUTINE CHECKS THE MISSILE EMPLACEMENT
C                       AGAINST GAME LIMITATIONS.
C*****
      SUBROUTINE SAMCHK (X,IERR)
      IERR=0
      IF (X.GE. 6000.) RETURN
      CALL ZRRMK (13)
      IERR=1
      RETURN
      END

```



```

C *****
C SUBROUTINE ERRMK      THIS SUBROUTINE SENDS PROMPTS AND ERROR MESSAGES
C                       TO THE USER.
C *****
C SUBROUTINE ERRMK (IERR)
C COMMON /MN/ IEAUL
C COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C COMMON /FAR/ X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
C 1) COMMON /FAR1/ XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
C COMMON /ERR/ MKEK1(3),MKEK2(15)
C IF (IERR.GT. 21) GO TO 210
-----
C FLASH THE ECRMPT WINDOW
C -----
CALL MIN (MINX2,MAXX2,2700.,185.,1)
CALL GSFRCE
GO TO (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
1180,190), IERR
10  WRITE (6,220)
GO TO 200
20  WRITE (6,230)
GO TO 200
30  WRITE (6,240)
GO TO 200
40  WRITE (6,250)
GO TO 200
50  WRITE (6,260)
GO TO 200
60  WRITE (6,270)
GO TO 200
70  WRITE (6,280)
GO TO 200
80  WRITE (6,290)
GO TO 200
90  WRITE (6,300)
GO TO 200
100 WRITE (6,310)
GO TO 200
110 WRITE (6,320)
GO TO 200
120 WRITE (6,330)
GO TO 200
130 WRITE (6,340)
GO TO 210
140 WRITE (6,350)
GO TO 210
150 WRITE (6,360)
GO TO 210
160 WRITE (6,370)
GO TO 210
170 WRITE (6,380)
GO TO 200
180 WRITE (6,390)
GO TO 200
190 WRITE (6,400)
GO TO 210
200 WRITE (6,410) X(MNUM),Y(MNUM),Z(MNUM),VEL(MNUM)
210 RETURN
220 FORMAT (25H MAXIMUM ERAKING EXCEEDED)
230 FORMAT (15H MAX G EXCEEDED)
240 FORMAT (12H ALT TOO LOW)
250 FORMAT (13H ALT TOO HIGH)
260 FORMAT (6H STALL)
270 FORMAT (15H PCF-UP TOO LOW)
280 FORMAT (13H BME DECF LOW)
290 FORMAT (12H BME DECF HI)
300 FORMAT (15H TOO FAR FM TGT)
310 FORMAT (17H HDG>5 DEG TO TGT)
320 FORMAT (19H FINAL BUN<2.33 SEC)
330 FORMAT (21H WC HCFICENTAL MOTICN)
340 FORMAT (17H TOO CLOSE TO TGT)
350 FORMAT (23H ENTER MISSILE LOCATION)
360 FORMAT (26H ENTER AIRCRAFT MILESTONES)
370 FORMAT (20H ENTER GUN LOCATIONS)

```



```
380 FCBMAT (18H MAX LIFT EXCEEDED)
390 FCBMAT (20H MAX THRUST EXCEEDED)
400 FCBMAT (28H X COORDINATE LESS THAN 6000)
410 FCBMAT (19H X,Y,Z,AND VELOCITY,4F9.1)
END
```



```

C *****
C SUBROUTINE VALSET THIS SUBROUTINE COMPUTES THE FLIGHT PARAMETERS FOR
C EACH MILESTONE. XDCT, YDOT, ZDOT, CA, HDG, AND RA
C ARE COMPUTED FOR OUTPUT TO P001 AND MICE II. TMD AND
C ACLIFT ARE COMPUTED FOR USE BY ERRCHK.
C *****
SUBROUTINE VALSET (IERR)
COMMON /PAR/ X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
1)
COMMON /PAR1/ XDCT(200),YDOT(200),ZDOT(200),MNUM,MBR
COMMON /PAR2/ T(200)
COMMON /PAR3/ TMD,ACLIFT,CLMAX,WL,TMAX,CDO,CDK,VMAX1,VMAX2
DIMENSION AXD(200),AYD(200),AZD(200),XDD(200),YDD(200),ZDD(200)
GEE=9.82
DX=X(MNUM)-X(MNUM-1)
DY=Y(MNUM)-Y(MNUM-1)
DZ=Z(MNUM)-Z(MNUM-1)
IF ((DX.NE. 0.) .OR. (DY.NE. 0.)) GO TO 10
IERR=12
RETURN
10 VAVGL=VAVG
DISTL=DIST
C -----
C LIMIT THE SPEED
C -----
DIST=SQRT(DX**2 + DY**2 + DZ**2)
IF ((VEL(MNUM) .GT. VMAX1) .AND. (MBR .EQ. 0)) VEL(MNUM)=VMAX1
IF ((VEL(MNUM) .GT. VMAX2) .AND. (MBR .GE. 1)) VEL(MNUM)=VMAX2
VAVG=(VEL(MNUM)+VEL(MNUM-1))/2.
DT=DIST/VAVG
T(MNUM)=T(MNUM-1)+DT
C -----
C COMPUTE AVERAGE VELOCITY COMPONENTS
C -----
AXD(MNUM)=VAVG*DX/DIST
AYD(MNUM)=VAVG*DY/DIST
AZD(MNUM)=VAVG*DZ/DIST
IF (MNUM .GE. 3) GO TO 40
C -----
C COMPUTE THE PARAMETERS FOR THE INITIAL LEG OF THE FLIGHT PATH
C -----
CA(1)=ATAN2(DZ,SQRT(DX**2 + DY**2))
IF ((DX.EQ. 0.) .OR. (DY.EQ. 0.)) GO TO 20
HDG(1)=ATAN2(DY,DX)
GO TO 30
20 IF ((DX.EQ. 0.) .AND. (DY .GT. 0.)) HDG(1)=1.57079
IF ((DX.EQ. 0.) .AND. (DY .LT. 0.)) HDG(1)=-1.57079
IF ((DY.EQ. 0.) .AND. (DX .GT. 0.)) HDG(1)=3.14159
IF ((DY.EQ. 0.) .AND. (DX .LT. 0.)) HDG(1)=0.
30 RA(1)=0.
XDCT(1)=VEL(1)*DX/DIST
YDOT(1)=VEL(1)*DY/DIST
ZDOT(1)=VEL(1)*DZ/DIST
RETURN
40 DELT=(T(MNUM)-T(MNUM-2))/2.
C -----
C COMPUTE THE ACCELERATION COMPONENTS
C -----
XDD(MNUM-1)=(AXD(MNUM)-AXD(MNUM-1))/DELT
YDD(MNUM-1)=(AYD(MNUM)-AYD(MNUM-1))/DELT
ZDD(MNUM-1)=(AZD(MNUM)-AZD(MNUM-1))/DELT
C -----
C WEIGHT AVERAGE OF THE VELOCITY COMPONENTS
C -----
HLAVG=DISTL/(DIST+DISTL)
TDX=(AXD(MNUM)/VAVG-AXD(MNUM-1)/VAVGL)*HDAVG+AXD(MNUM-1)/VAVGL
TDY=(AYD(MNUM)/VAVG-AYD(MNUM-1)/VAVGL)*HDAVG+AYD(MNUM-1)/VAVGL
TDZ=(AZD(MNUM)/VAVG-AZD(MNUM-1)/VAVGL)*HDAVG+AZD(MNUM-1)/VAVGL
C -----
C MAKE THE WEIGHTED VALUES A UNIT VECTOR AND COMPUTE THE MILESTONE
C VELOCITY COMPONENTS
C -----
UNIT=SQRT(TDX*TDX + TDY*TDY + TDZ*TDZ)
IF (ABS(UNIT) .GE. 0.01) GO TO 50
IERR=12

```



```

50  RETURN
    TDX=TDX/UNIT
    TDY=TDY/UNIT
    TDZ=TDZ/UNIT
    XDCT(MNUM-1)=VEL(MNUM-1)*TDX
    YDOT(MNUM-1)=VEL(MNUM-1)*TDY
    ZDOT(MNUM-1)=VEL(MNUM-1)*TDZ
    IF (SQRT(TDX*TDX+TDY*TDY) .GE. 0.01) GO TO 60
    CA(MNUM-1)=1.5533
    GC TO 70
60  CA(MNUM-1)=ATAN2(IDZ,SQRT(TDX*TDX+TDY*TDY))
70  IF ((TDX .EQ. 0.) .OR. (TDY .EQ. 0.)) GO TO 80
    HDG(MNUM-1)=ATAN2(TDY,TDX)
    GO TO 90
80  IF ((TDX .EQ. 0.) .AND. (TDY .GT. 0.)) HDG(MNUM-1)=1.57079
    IF ((TDX .EQ. 0.) .AND. (TDY .LT. 0.)) HDG(MNUM-1)=-1.57079
    IF ((TDY .EQ. 0.) .AND. (TDX .LT. 0.)) HDG(MNUM-1)=3.14159
    IF ((TDY .EQ. 0.) .AND. (TDX .GT. 0.)) HDG(MNUM-1)=0.
-----
C  COMPONENTS OF ACCELERATION IN THE AIRCRAFT COORDINATE SYSTEM
C  -----
90  CASIN=SIN(CA(MNUM-1))
    CACOS=COS(CA(MNUM-1))
    HDSIN=SIN(HDG(MNUM-1))
    HDCOS=COS(HDG(MNUM-1))
    A11=XDD(MNUM-1)*HDCOS*CACOS
    A21=-XDD(MNUM-1)*HDSIN
    A31=-XDD(MNUM-1)*HDCOS*CASIN
    A12=YDD(MNUM-1)*HDSIN*CACOS
    A22=YDD(MNUM-1)*HDCOS
    A32=-YDD(MNUM-1)*HDSIN*CASIN
    A13=(ZDD(MNUM-1)+GEE)*CASIN
    A23=0.
    A33=(ZDD(MNUM-1)+GEE)*CACOS
-----
C  COMPUTE ACCELERATION PARALLEL AND PERPENDICULAR TO FLIGHT PATH
C  -----
    TMC=(A11+A12+A13)/GEE
    ACLIFT=(SQRT((A21+A22)**2+(A31+A32+A33)**2))/GEE
    IF (ABS(A31+A32+A33) .GE. 0.01) GO TO 100
    RA(MNUM-1)=1.57079
    GO TO 110
100  RA(MNUM-1)=ATAN2((A21+A22),(A31+A32+A33))
110  CALL ERRMK(22)
    RETURN
    END

```



```

C*****
C SUBROUTINE ERRCHK
C      THIS SUBROUTINE EVALUATES THE FLIGHT PARAMETERS TO
C      ENSURE THAT THEY MEET THE DESIRED SIMULATION LIMITS.
C*****
      SUBROUTINE ERRCHK (IERR)
      COMMON /PAR/ X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
1)    COMMON /PAR1/ XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
      COMMON /PAR2/ T(200)
      COMMON /PAR3/ TARGX,TARGY
      COMMON /PAR3/ TMD,ACLIFT,CLMAX,WL,TMAX,CD0,CDK
      COMMON /PAR4/ APPMAX,HTMIN,HTMAX,SPDMIN,GMAX,POPALN
      DATA TMSAV/-1./
      DX=TARGX-X(MNUM)
      DY=TARGY-Y(MNUM)
      DIST=SQRT(DX**2 + DY**2)
C-----
C INITIALIZE POPALN AND IMSAV
C-----
      IF ((MNUM.NE.2) .AND. (TMSAV.NE.-1.)) GO TO 10
      PCFALT=0.
      TMSAV=TMAX
10     IF (MBR.EQ.0 .AND. TMAX.NE.TMSAV) TMAX=IMSAV
      CALL ERRMK (23)
      IF (MBR.GE.1) GO TO 20
      IF ((DIST.LT. PCFMIN).OR. (Z(MNUM) .LT. APPMAX)) GO TO 20
      IERR=4
      RETURN
20     IF (DIST.LE. PCFMIN .AND. Z(MNUM).GT. POPALT) POPALT=Z(MNUM)
      IF (Z(MNUM) .GE. HTMIN) GO TO 30
      IERR=3
      RETURN
30     Z(MNUM)=HTMIN
      IF (Z(MNUM) .LE. ETMAX) GO TO 40
      IERR=4
      RETURN
40     Z(MNUM)=HTMAX
      IF (VEL(MNUM) .GE. SPDMIN) GO TO 50
      IERR=5
      RETURN
50     IF (MNUM.EQ.2) RETURN
      IF (ACLIFT.LE. GMAX) GO TO 60
      IERR=2
      RETURN
C-----
C COMPUTE DRAG AND LIFT FORCES
C-----
60     RH0=0.0256*EXP(-0.103*(Z(MNUM)/1000.))
      IF (Z(MNUM) .GE. 10670.) GO TO 70
      RH0=0.00793*EXP(-0.156*(Z(MNUM)-10670.)/1000.)
70     CL=2*ACLIFT/(RH0*(VEL(MNUM)**2)/WL)
      IF (CL.LE. CLMAX) GO TO 80
      IERR=17
      RETURN
80     DW=((CL0+CDK*CL*CL)/(CL)*ACLIFT
      TW=TMD+DW
      IF ((TW.LE. TMAX) .AND. (TW.GE.0.)) GO TO 90
      IF (TW.GT. TMAX) IERR=18
      DW=(RH0*(VEL(MNUM)**2)/WL)*(0.05+CD0+CDK*(CL**2))/2.
      IF (TMD+DW.LE.0) IERR=1
      IF (IERR.NE.0) RETURN
90     IF (MNUM.NE. MBR) RETURN
C-----
C EVALUATE BECOMING RUN
C-----
      DT=T(MNUM)-T(MNUM-1)
      TGTHDG=ATAN2(DY,DX)*57.29578
      IF (TGTHDG.LT.0.) TGTHDG=TGTHDG+360.
      DX=X(MNUM)-X(MNUM-1)
      DY=Y(MNUM)-Y(MNUM-1)
      ACHDG=ATAN2(DY,DX)*57.29578
      IF (ACHDG.LT.0.) ACHDG=ACHDG+360.
      HDGLMT=ABS(TGTHDG-ACHDG)
      IF (POPALT.GE.1000.) GO TO 100
      IERR=6

```



```

100  RETURN
      IF (Z(MNUM) .GE. 100.) GO TO 110
      IERR=7
      RETURN
110  IF (Z(MNUM) .LE. 2000.) GO TO 120
      IERR=8
      RETURN
120  IF (DIST .LE. 2000.) GO TO 130
      IERR=9
      RETURN
130  IF (HDGLMT .LE. 5.) GO TO 140
      IERR=10
      RETURN
140  IF (DT .GE. 2.33) GO TO 150
      IERR=11
150  TMAX=1.2*TMAX
      RETURN
      END

```



```

C*****
C SUBROUTINE PTHPLT THIS SUBROUTINE NOTIFIES THE USER THAT HIS MILE-
C STONE HAS BEEN ACCEPTED AND THEN SAVES THE SUCCESSFUL
C PCINT IN A TEMPORARY DISK FILE.
C*****
SUBROUTINE PTHPLT
COMMON /PAR/ X(200),Y(200),Z(200),HDG(200),CA(200),PA(200),VEL(200)
1)
COMMON /PAR1/ XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
COMMON /MN/ IBAUD
COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
REAL*4 TX1,TY1,IX2,IY2
LOGICAL *1CHARS(12)
TX1=X(MNUM-1)
TX2=Y(MNUM)
TY1=Y(MNUM-1)
TY2=Y(MNUM)

C-----
C IDENTIFY THE MAP, IFAW AND LABEL THE FLIGHT LEG
C-----
CALL WIN (MINX,MAXX,18000.,12000.,0)
CALL GBMCVE (TX1,TY1)
CALL GBDEAW (TX2,IY2)
IF (MNUM.GT.99) IW=3
IF ((MNUM.GT.9) .AND. (MNUM.LT.100)) IW=2
IF (MNUM.LE.9) IW=1
CALL GAXITC (1,MNUM,IW,CHARS)
CALL GBCHAR (TX2,IY2,2.,0.,IW,CHARS)
CALL GSFRCE

C-----
C SAVE THE PCINT IN 'TEMP DATA'
C-----
WRITE (19,10) X(MNUM),Y(MNUM),Z(MNUM),VEL(MNUM),MBR
RETURN
10 FORMAT (4F10.0,I3)
END

```

```

C*****
C SUBROUTINE AIMPT THIS SUBROUTINE MARKS A LINE FROM THE CURRENT
C AIRCRAFT POSITION TO THE TARGET AS A USER AID IN
C ALIGNING BOMB RELEASE POINT.
C*****
SUBROUTINE AIMPT (I)
COMMON /IAR/ IARGX,IARGY
COMMON /PAR/ X(200),Y(200)
REAL*4 TX,TY,IX1,IY1
TX=X(I)
TY=Y(I)
TX1=IARGX
TY1=IARGY
CALL GSLT (1)

C-----
C MOVE TO CURRENT MILESTONE COORDINATES
C-----
CALL GBMCVE (TX,TY)

C-----
C DRAW DASHED LINE TO TARGET
C-----
CALL GBDEAW (TX1,TY1)
CALL GSLT (0)
RETURN
END

```



```

C*****
C SUBROUTINE ELFIN          THIS SUBROUTINE REQUESTS THE USER OPTIONS AND CLOSES
C                           THE GRAPHIC SUBROUTINES.
C*****
C SUBROUTINE ELFIN (LAST)
C   COMMON /CFT/ IGUN,IPNCH,TEXT,ISAM,IMP,KER
C   COMMON /MM/ IEAUD
C   COMMON /MM1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C-----
C CLOSE GRAPHICS ROUTINES
C-----
C   CALL DSIFRM
C-----
C OLD (LAST=0) OR NEW (LAST=1) FLIGHT PATH?
C-----
C   IF (LAST.EQ. 0) GO TO 20
C-----
C REQUEST USER OUTPUT OPTIONS
C-----
10  WRITE (6,40)
    READ (5,*) LAST
    CALL FRTCMS ('CLASCRN ')
    IF (LAST.GE. 2 .OR. LAST.LT.0) GO TO 10
    LAST=1+82*LAST
20  WRITE (6,50)
    WRITE (6,60)
    READ (5,*) IPNCH
    CALL FRTCMS ('CLASCRN ')
C-----
C IF MILESTONES CAME FROM DISK, RETURN END OF FILE MARKER
C-----
    IF (LAST.EQ. 0) LAST=83
    IF (IPNCH.EQ. 0) RETURN
    IF (IPNCH.EQ. 2) GO TO 30
    WRITE (6,70)
    READ (5,*) TEXT
    CALL FRTCMS ('CLASCRN ')
    IF (IPNCH.EQ. 1) RETURN
30  WRITE (6,80)
    READ (5,*) ISAM
    CALL FRTCMS ('CLASCRN ')
    IF ((ISAM.LT. 1) .OR. (ISAM.GT. 7)) GO TO 30
    RETURN
40  FORMAT (51H ARE YOU FINISHED WITH THIS FLIGHTPATH: 0=NO; 1=YES)
50  FORMAT (43H OUTPUT FILE: 0=NO OUTPUT; 1=POOL FILE ONLY)
60  FORMAT (51H          2=MICE FILE ONLY; 3=POOL & MICE FILES)
70  FORMAT (49H EXTENDED OUTPUT: 0=NOT WANTED; 1=EXTENDED OUTPUT)
80  FORMAT (29H MISSILE TYPE BETWEEN 1 AND 7)
END

```



```

C *****
C SUBROUTINE PRESET
C      THIS SUBROUTINE WRITES THE P001 AND MICE II FILES
C      FOR THE USER.
C *****
C SUBROUTINE PRESET
C COMMON /PAR/ X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),VEL(200)
C 1) COMMON /PAR1/ XDOT(200),YDOT(200),ZDOT(200),MNUM,MBR
C COMMON /PAR2/ T(200),XGUN(7),YGUN(7),ZGUN(7),XSAM,YSAM,ZSAM
C COMMON /OPT/ IPNCH,IPNCH,TEXT,ISAM,IME,KER
C COMMON /NAM/ INAM1(2),INAM2(2),TUNK,IANE,IDEST
C DIMENSION X10(200),Y10(200)
C DIMENSION RCSTAB(1000),TINK(10)
C DIMENSION VAT1N2(208),VAT3(208),VAT5(208)
C -----
C VULNERABLE AREA TABLE VS TYPE 1 AND 2 WEAPONS
C -----
C DATA VAT1N2/2*.4645,6*7.107,2*.6568,6*5.551,2*.6968,6*5.574,2*.6568,
18*.67.357,2*.6968,6*5.574,2*.6568,6*5.551,2*.6968,6*5.574,2*.6568,
26*.7.357,2*.6968,6*5.574,2*.4645,6*7.432,2*.6568,6*2.358,2*.4645,6*
33.298,2*.6568,6*2.358,2*.4645,6*7.432,2*.6568,6*2.358,2*.4645,6*3.
4298,2*.6568,6*2.358,2*.4645,6*5.551,2*.6968,6*5.574,2*.6568,6*7.35
57,2*.6968,6*5.574,2*.6568,6*5.551,2*.6968,6*5.574,2*.6568,6*7.357,
62*.6968,6*5.574,2*.4645,6*7.107/
C -----
C VULNERABLE AREA TABLE VS TYPE 3 WEAPONS
C -----
C DATA VAT3/2*.12.54,6*13.47,2*.9.553,6*10.51,2*.9.639,6*11.15,2*.12.64,
16*.14.78,2*.9.639,6*11.15,2*.9.853,6*10.51,2*.9.639,6*11.15,2*.12.64,6*
214.78,2*.9.639,6*11.15,3*.1.394,2*.4.762,6*6.240,2*.5.342,6*7.432,2*.4.
3762,6*6.240,3*.1.394,6*.4.762,6*6.240,2*.5.342,6*7.432,2*.4.762,6*6.24
40,2*.9.639,6*10.51,2*.9.639,6*11.15,2*.12.64,6*14.78,2*.9.639,6*11.15,
52*.9.639,6*10.51,2*.9.639,6*11.15,2*.12.64,6*14.78,2*.9.639,6*11.15,2*
612.54,6*13.47/
C -----
C VULNERABLE AREA TABLE VS TYPE 5 WEAPONS
C -----
C DATA VAT5/8*35.37,3*43.22,3*47.10,6*62.53,6*47.10,3*43.22,3*47.10,
18*62.53,3*47.10,6*5.761,3*27.43,6*33.07,3*27.45,3*5.761,3*27.43,3*
233.07,3*27.45,6*5.761,3*47.10,6*62.53,6*47.10,3*43.22,3*47.10,6*62
3.53,6*47.10,3*55.37/
C -----
C BALAB CROSS SECTION TABLE
C -----
C DATA RCSTAB/19*1000.,19*100.,19*10.,19*10.,19*10.,19*100.,19*1000.
1,867*0.0/
C -----
C CALL ERRSET SUPPRESSES ANY POSSIBLE UNDERFLOW PROBLEMS THAT MAY
C RESULT FROM MANIPULATION OF SCENARIO PARAMETERS.
C -----
C CALL ERRSET (208,50,-1,1,1)
C JAM=0
C -----
C CABD 2 TIME INCREMENT CALCULATION
C -----
C TINC=T(MNUM)/1000.+0.0008
C -----
C CABD 6 TIME INCREMENT CALCULATIONS
C -----
C TINKI=0
C DO 10 I=1,9
C TINK(I)=TINKI+T(MNUM)/10
C TINKI=TINK(I)
10 CONTINUE
C -----
C /////***** FUNCH PROGRAM*****/////
C -----
C OPTION TO PUNCH THE P001 CARD DECK OR MICE-II CARD DECK
C -----
C IF (IPNCH.EQ.2) GO TO 30
C -----
C COMMENCE PUNCHED OUTPUT OF THE P001 CARD DECK.
C -----

```



```

C THE JCL CARDS.
C-----
WRITE (18,60) INAM1(1), INAM1(2), IUNR, IANR, INAM1(1), INAM1(2), IUNR
IF (IEXT.EQ.2) WRITE (18,70) IUNR
WRITE (18,80)
WRITE (18,90)
WRITE (18,100)
WRITE (18,110)
WRITE (18,120)
WRITE (18,130)
WRITE (18,140)
WRITE (18,150)
WRITE (18,160)
C-----
C LEADING BLANK DATA CARD SIGNIFIES RADAR MASKING ANGLE OF ZERO.
C-----
C THE OUTPUT TITLE CARD.
C-----
WRITE (18,190)
C-----
C CARD 2
C-----
WRITE (18,170)
WRITE (18,200) I(MNUM), TIME
C-----
C THE 2A CARDS (MILESTONES).
C-----
DO 20 I=1, INUM
WRITE (18,210) T(I), X(I), Y(I), Z(I), XDOT(I), YDOT(I), ZDOT(I), HDG(I),
1CA(I), SA(I)
20 CONTINUE
WRITE (18,230)
C-----
C CARD 3 (GUN EMPLACEMENT CARD).
C-----
WRITE (18,240) XGUN(1), YGUN(1), ZGUN(1)
C-----
C CARD 4 (GUN TYPE).
C-----
WRITE (18,220)
WRITE (18,250)
C-----
C CARD 5
C-----
WRITE (18,180)
WRITE (18,260)
C-----
C CARD 6
C-----
WRITE (18,270)
WRITE (18,280) (LINK(I), I=1,9)
C-----
C CARD 7 (VULNERABLE AREA TABLE VS TYPE 1 AND 2 WEAPONS)
C-----
WRITE (18,290)
WRITE (18,300)
WRITE (18,310) (VAT1N2(I), I=1,208)
C-----
C CARD 12 (EXECUTE RUN).
C EXTENDED OUTPUT OPTION
C-----
WRITE (18,320)
IF (IEXT.NE.1) WRITE (18,340)
IF (IEXT.EQ.1) WRITE (18,330)
C-----
C THE REMAINDER OF THE CARDS INTRODUCE NEW GUN LOCATIONS, GUN TYPES
C AND VULNERABLE AREA TABLES TO BE EXECUTED BY THE PROGRAM.
C-----
WRITE (18,230)
WRITE (18,240) XGUN(2), YGUN(2), ZGUN(2)
C-----
C EXTENDED OUTPUT OPTION
C-----
WRITE (18,320)
IF (IEXT.NE.1) WRITE (18,340)

```



```

IF (TEXT.EQ.1) WRITE (18,330)
WRITE (18,230)
WRITE (18,240) XGUN(3), YGUN(3), ZGUN(3)
WRITE (18,220)
WRITE (18,350)
C-----
C EXTENDED OUTPUT OPTION
C-----
WRITE (18,320)
IF (TEXT.NE.1) WRITE (18,340)
IF (TEXT.EQ.1) WRITE (18,330)
WRITE (18,230)
WRITE (18,240) XGUN(4), YGUN(4), ZGUN(4)
C-----
C EXTENDED OUTPUT OPTION
C-----
WRITE (18,320)
IF (TEXT.NE.1) WRITE (18,340)
IF (TEXT.EQ.1) WRITE (18,330)
WRITE (18,230)
WRITE (18,240) XGUN(5), YGUN(5), ZGUN(5)
WRITE (18,220)
WRITE (18,350)
C-----
C CARD 7 (VULNERABLE AREA TABLE VS TYPE 3 WEAPONS)
C-----
WRITE (18,290)
WRITE (18,400)
WRITE (18,310) (VAT3(I), I=1,208)
C-----
C-----
C EXTENDED OUTPUT OPTION
C-----
WRITE (18,320)
IF (TEXT.NE.1) WRITE (18,340)
IF (TEXT.EQ.1) WRITE (18,330)
WRITE (18,230)
WRITE (18,240) XGUN(6), YGUN(6), ZGUN(6)
WRITE (18,220)
WRITE (18,370)
C-----
C EXTENDED OUTPUT OPTION
C-----
WRITE (18,320)
IF (TEXT.NE.1) WRITE (18,340)
IF (TEXT.EQ.1) WRITE (18,330)
WRITE (18,230)
WRITE (18,240) XGUN(7), YGUN(7), ZGUN(7)
WRITE (18,220)
WRITE (18,380)
C-----
C CARD 7 (VULNERABLE AREA TABLE VS TYPE 5 WEAPON)
C-----
WRITE (18,290)
WRITE (18,410)
WRITE (18,310) (VAT5(I), I=1,208)
C-----
C EXTENDED OUTPUT OPTION
C-----
WRITE (18,320)
IF (TEXT.NE.1) WRITE (18,340)
IF (TEXT.EQ.1) WRITE (18,330)
WRITE (18,390)
IF (IPNCH.EQ.1) RETURN
C-----
C FORMAT STATEMENTS
C-----
30 CONTINUE
C-----
C COMMENCE PUNCHED OUTPUT OF THE MICE-II CARD DECK
C-----
C CHECK FOR VALID MISSILE DESIGNATION NUMBER FOR MICE II
IF (ISAM.LE.7.AND.ISAM.GE.1) GO TO 40
RETURN
C-----
C THE JCL CARDS

```



```

C-----
40  WRITE (17,420) INAM2(1),INAM2(2),IUNB,IANR,INAM2(1),INAM2(2),IUNR
    IF (IDEST.EQ.2) WRITE (17,430) IUNR
    WRITE (17,440)
    WRITE (17,450)
    WRITE (17,460)
    WRITE (17,470)
C-----
C THE OUTPUT TITLE CARDS
C-----
    WRITE (17,480)
    WRITE (17,490)
C-----
C THE PROBLEM RUN INPUT CARDS, CHECKING FOR TYPE OF MISSILE
C TC INPUT TO MICE II
C-----
    WRITE (17,500)
    IF (ISAM.EQ.1) WRITE (17,570)
    IF (ISAM.EQ.2) WRITE (17,580)
    IF (ISAM.EQ.3) WRITE (17,590)
    IF (ISAM.EQ.4) WRITE (17,600)
    IF (ISAM.EQ.5) WRITE (17,610)
    IF (ISAM.EQ.6) WRITE (17,620)
    IF (ISAM.EQ.7) WRITE (17,630)
C-----
C THE PSSK CALCULATION CARDS
C-----
    WRITE (17,510)
    WRITE (17,520)
C-----
C THE RADAR CROSS SECTION TABLE
C-----
    WRITE (17,530)
    WRITE (17,540)
    WRITE (17,550)
    WRITE (17,560)
    WRITE (17,570) (RCSTAB(I),I=1,133)
C-----
C THE SIMULATION TIME PARAMETERS
C-----
    WRITE (17,580)
    WRITE (17,590)
C-----
C THE MISSILE LAUNCHER LOCATION
C-----
    PTFAC=3.28084
    WRITE (17,600)
    XFSAM=XSAM*PTFAC
    YFSAM=YSAM*PTFAC
    ZFSAM=ZSAM*PTFAC
    WRITE (17,610) XFSAM,YFSAM,ZFSAM
C-----
C THE TARGET TRAJECTORY
C-----
    WRITE (17,620)
    WRITE (17,630) MNUM
C-----
C THE MILESTONE CARDS
C-----
    DO 50 I=1,MNUM
    X(I)=X(I)*PTFAC
    Y(I)=Y(I)*PTFAC
    Z(I)=Z(I)*PTFAC
    VEL(I)=VEL(I)*PTFAC
    WRITE (17,640) I(I),X(I),Y(I),Z(I),VEL(I),CA(I),RA(I),HDG(I)
50  CCNTINUE
C-----
C END OF PUNCHED CARDS
C-----
    WRITE (17,650)
    WRITE (17,660)
C-----
    RETURN
C-----
C /////***** FORMAT STATEMENTS *****/////
C-----

```



```

60  FORMAT (2H//,2A4,6H JOB (,I4,1H,,I4,2H),,1H',2A4,1H ,I4,2HP')
70  FORMAT (19H// *MAIN ORG=NP GVM1.,I4,1HP)
80  FORMAT (16H// EXEC PGM=PIAD)
90  FORMAT (42H// STEPLIB DD DISP=SHR,DSN=MSS.F0559.PIPSAV)
100 FORMAT (47H// GC.FT07F001 DD UNIT=SYSDA,SPACE=(CYL,(1,1),),)
110 FORMAT (41H// CC= (RECFM=VBS, LRECL=404, BLKSIZE=3236))
120 FORMAT (22H// GC.FT09F001 DD DUMMY)
130 FORMAT (25H// GC.FT06F001 DD SYSOUT=A)
140 FORMAT (18H// GC.FTCSF001 DD *)
150 FORMAT (11H1,1,J,C,C,C)
160 FORMAT (2H01)
170 FORMAT (2H02)
180 FORMAT (2H05)
190 FORMAT (42H AIRCRAFT COMBAT SURVIVABILITY SCENARIO)
200 FORMAT (7H0,12,0,,F7.2,9H,100000.,F7.4,1H/)
210 FORMAT (10(F7.1,1X))
220 FORMAT (2H04)
230 FORMAT (2H03)
240 FORMAT (3(1X,F7.C),1CH 0.0,360.0)
250 FORMAT (19H0,1,1,1,1,1,0.0,50.)
260 FORMAT (7H1,1,1.0)
270 FORMAT (2H06)
280 FORMAT (3H1,9,9(1X,F7.3))
290 FORMAT (2H07)
300 FORMAT (49H VULNERABLE AREA TABLE VS TYPE 1 AND 2 WEAPONS)
310 FORMAT (8F8.3)
320 FORMAT (2H12)
330 FORMAT (13H1,1,1,1,1,1,1,1)
340 FORMAT (13H0,0,0,0,1,1,1,1)
350 FORMAT (20H0,2,1,1,1,1,1,0.0,50.0)
360 FORMAT (2CH0,3,4,4,1,1,0.0,50.0)
370 FORMAT (2CH0,3,3,4,1,1,0.0,50.0)
380 FORMAT (2CH0,5,3,2,1,1,0.0,50.0)
390 FORMAT (2H/*)
400 FORMAT (43H VULNERABLE AREA TABLE VS TYPE 3 WEAPONS)
410 FORMAT (42H VULNERABLE AREA TABLE VS TYPE 5 WEAPONS)
420 FORMAT (2H//,2A4,6H JOB (,I4,1H,,I4,2H),,1H',2A4,1H ,I4,2HP')
430 FORMAT (19H// *MAIN ORG=NP GVM1.,I4,1HP)
440 FORMAT (28H// EXEC PGM=PIAD, REGION=180K)
450 FORMAT (44H// STEPLIB DD DSN=MSS.F0559.MICESAV, DISP=SHR )
460 FORMAT (22H// FT06F001 DD SYSCUT=A)
470 FORMAT (15H// FTCSF001 DD *)
480 FORMAT (2H01)
490 FORMAT (55HAE 3251 *****SURVIVABILITY SCENARIO***** SAM ENCOUNTER
1)
500 FORMAT (2H02)
510 FORMAT (2H06)
520 FORMAT (9X,1H0,8X,2H0.,9X,1H0,4X,1H0,4X,1H0,8X,2H0.,8X,2H0.)
530 FORMAT (2H08)
540 FORMAT (7X,3H23.,3X,2H19,4X,1H7,4X,1H1)
550 FORMAT (8X,2H0.,7X,3H10.,7X,3H20.,7X,3H30.,7X,3H40.,7X,3H50.,7X,3H
160.,7X,3H70.,7X,3H80.,7X,3H90.,6X,4H100.,6X,4H110.,6X,4H120.,6X,
24H130.,6X,4H140.,6X,4H150.,6X,3H160,6X,4H170.,6X,4H180.)
560 FORMAT (6X,4H-90.,6X,4H-60.,6X,4H-30.,8X,2H0.,7X,3H30.,7X,3H60.,7X
1,3H90.)
570 FORMAT (7F10.2,/,7F10.2,/,5F10.2)
580 FORMAT (2H09)
590 FORMAT (7X,3H0.5,7X,3H0.1,7X,3H8.0,7X,3H0.0,8X,2H0.,7X,3H30.,8X,2H
10.)
600 FORMAT (2H11)
610 FORMAT (2X,F8.2,2F10.2)
620 FORMAT (2H12)
630 FORMAT (9X,1H1,2X,I3,4X,1H1,4X,1H0,8X,2H0.,3X,2H0.,3X,2H0.)
640 FORMAT (F7.1,4F10.1,3F6.1,4X,2H0.,4X,2H0.)
650 FORMAT (2H20)
660 FORMAT (2H/*)
670 FORMAT (9X,1H1,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
680 FORMAT (9X,1H2,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
690 FORMAT (9X,1H3,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
700 FORMAT (9X,1H4,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
710 FORMAT (9X,1H5,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
720 FORMAT (9X,1H6,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
730 FORMAT (9X,1H7,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
END

```



```

C *****
C BLOCK DATA
C *****
      BLOCK DATA
COMMON /MN1/  MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
COMMON /ERR/  MKERX(3),MKERY(20)
COMMON /PAR/  X(200),Y(200),Z(200),HDG(200),CA(200),RA(200),TEL(200)
1)
COMMON /PAR1/ XDCI(200),YDOT(200),ZDOT(200),ANUM,MBR
COMMON /PAR2/ TI(200),XGUN(7),YGUN(7),ZGUN(7),XSA4,YSAM,ZSAM,GB(7)
COMMON /PAR3/ TMD,ACLIFT,CLMAX,WL,TMAX,CDO,CDK,VMAX1,VMAX2
COMMON /PAR4/ APFMAX,HTMIN,HTMAX,SPDMIN,GMAX,POPMIN
COMMON /TAR/  TARGX,TARGY
C -----
C MAP WINDOW X COORDINATES
C -----
      DATA MINX/100/
      DATA MAXX/3700/
C -----
C ALTIMETER WINDOW X COORDINATES
C -----
      DATA MINX1/3725/
      DATA MAXX1/3925/
C -----
C FRECHTING WINDOW X COORDINATES
C -----
      DATA MINX2/3950/
      DATA MAXX2/4000/
C -----
C COMMON Y COORDINATES FOR THE THREE WINDOWS
C -----
      DATA MINY/400/
      DATA MAXY/2900/
C -----
C EBCB MESSAGE COORDINATES
C -----
      DATA MKERX/640,340,10/
      DATA MKERY/160,140,120,100,80,60,160,140,120,100,80,60,160,140,120
1,100,80,60,180,40/
C -----
C PREDETERMINED WEAPON EMPLACEMENT PARAMETERS
C -----
      DATA XGUN/14800.,16200.,13600.,13400.,11300.,15600.,12800./
      DATA YGUN/9000.,3200.,7200.,3600.,9700.,10900.,7500./
      DATA ZGUN/40.,40.,20.,20.,50.,90.,20./
      DATA XSA4/12000./,YSAM/6000./,ZSAM/50./
      DATA GB/1000.,1000.,1400.,1400.,2500.,2500.,2500./
C -----
C SIMULATION PARAMETERS
C -----
      DATA CLMAX/1./,WL/100./,SPDMIN/90./,GMAX/6./,HTMIN/60./
      DATA HTMAX/2050./,APFMAX/457./,TMAX/0.4./,CDO/0.015/,CDK/0.1/
      DATA VMAX1/260./,VMAX2/310./,FCPMIN/6000./
      DATA TARGX/14000./,TARGY/7220./
      END

```


APPENDIX G

PROPOSED IBMPIP LISTING

```

C*****
C MAIN PROGRAM 16/MAB/84 VERSION
C CNTRCLS THE EXECUTION OF THE ENTIRE PROGRAM.
C THIS IS THE PROPOSED MODIFICATION OF THE JOHNS'
C IT WILL WORK FOR SOME CASES BUT NOT FOR EVERY
C CASE. THE FLIGHTPATH PACKAGE (3IGCAL PACKAGE)
C IS ALSO LISTED SEPARATELY AS T11 FORTRAN. THE
C T11 PROGRAM IS SET UP TO WRITE TO FILE AND
C DOES NOT CONTAIN ANY GRAPHICS DEPENDENT CALLS.
C -- D.A.V. 16 MAR84 --
C*****
COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
COMMON /LCC/ X1,Y1,Z1,V1,LR,LR2
COMMON /MN/ IEAUC
COMMON /ABC1/ AX(1000),AY(1000),AZ(1000),WU(1000),WV(1000),WW(1000)
1) ,WH(1000),WP(1000),WR(1000),WG(1000),I(1000),ICT
COMMON /ABC2/ XNEW,XTEMP,XOLD,YNEW,YTEMP,YOLD,ZNEW,ZTEMP,ZOLD
COMMON /AED2/ VSTALL,VMAX,VMAX1,VMAX2,DELTAZ,DELTAZ,DELTAZ
COMMON /FAR2/ XGUN(7),YGUN(7),ZGUN(7),XSAM,YSAM,ZSAM,GR(7)
COMMON /CPT/ IGUN,LENCH,LEXT,ISAM,IME,KER,IMPZ
COMMON /NAM/ INAM1(2),INAM2(2),IUNR,IANR,IDEST
COMMON /TAR/ TARGX,TARGY
CALL ERRSET (207,256,-1,1,209)
ICT=2
VMAX=VMAX1
CALL BEGIN
CALL SCENE (0,0)
REWIND 1C
C-----
C DRAW THE FIXED GUN SITE & ITS ENGAGEMENT CIRCLE
C-----
CALL GUNLOC (XGUN(7),YGUN(7),GR(7))
C-----
C IGUN=1==> READ USER DEFINEABLE GUN SITES FROM TERMINAL
C-----
IF (IGUN.NE.1) GO TO 40
C-----
C READ COMMENTS IN WINDOW: GRAPHICS INITIALIZER
C-----
CALL WIN (MINX,MAXX,18000.,12000.,0)
C-----
C GSLT DEFINES THE TYPE OF LINE TO BE DRAWN: BOLD, DASHED ETC.
C-----
CALL GSLT (1)
C-----
C GEGARC DRAWS ARCS AND IF SPECIFIED, WHOLE CIRCLES
C-----
CALL GEGARC (3000.,3000.,TARGX,TARGY,1.,1.,2.)
CALL GSLT (0)
CALL GSFRCE
IEER=16
CALL ERRMK (IEER)
DC 40 I=1,0
10 CONTINUE
IEER=0
IF (IMPZ.NE.1) GO TO 20
CALL XYPT
ZGUN(I)=20.
GC TO 30
20 CONTINUE
CALL XYPT
CALL ZVPI
ZGUN(I)=21
30 CONTINUE
XGUN(I)=X1
YGUN(I)=Y1
C-----
C IF GUN TYPE 3, CHK FOR 3 KM SEPARATION FM BRIDGE
C-----
IF (I.GE.5) CALL GUNCHK (X1,Y1,IEER)
IF (IEER.NE.0) GO TO 10
C-----
C DRAW GUN SITE AND ENGAGEMENT CIRCLE OF SELECTED GUNSITES
C-----
CALL GUNLOC (XGUN(I),YGUN(I),GR(I))
WRITE (10,250) XGUN(I),YGUN(I),ZGUN(I)

```



```

40      CONTINUE
C-----
C READ USER DEFINEABLE SAM SITE FM TERMINAL
C-----
50      CONTINUE
      IERR=14
      CALL ZERRMK (IERR)
      IF (IMPZ.NE.1) GO TO 60
      CALL XYPT
      ZSAM=20.
      GO TO 70
60      CCNTINUE
      CALL XYPT
      CALL ZVPT
      ZSAM=21
70      CCNTINUE
      XSAM=X1
      YSAM=Y1
C-----
C VERIFY SAM SITE AT LEAST 6KM FM WESTERN BOUNDARY
C-----
      CALL SAMCHK (X1,IERR)
      IF (IERR.NE.3) GO TO 50
C-----
C DRAW SAM SITE
C-----
      CALL GUNLCC (XSAM,YSAM,150.)
      WRITE (10,250) XSAM,YSAM,ZSAM
      GO TO 120
C-----
C IGUN=2==> READ GUN SITES FROM DATA FILE
C-----
80      CONTINUE
      IF (IGUN.NE.3) GO TO 100
      DO 90 I=1,6
      READ (10,250) XGUN(I),YGUN(I),ZGUN(I)
90      CONTINUE
      READ (10,250) XSAM,YSAM,ZSAM
C-----
C DRAW GUN SITES AND ENGAGEMENT CIRCLES
C-----
100     CCNTINUE
      DO 110 I=1,6
      CALL GUNLCC (XGUN(I),YGUN(I),ZF(I))
110     CONTINUE
      CALL GUNLCC (XSAM,YSAM,150.)
120     CONTINUE
C-----
C THIS SECTION ACCEPTS THE FLIGHT MILESTONES
C-----
      IERR=15
      CALL ZERRMK (IERR)
      REWIND 11
C-----
C READ MILESTONES FROM THE DISK
C-----
130     IF (IMP.NE.0) GO TO 140
      READ (11,260) T(ICT),TX(ICT),TY(ICT),WZ(ICT),WU(ICT),WV(ICT),WW(IC
1T),WH(ICT),WP(ICT),WA(ICT),WG(ICT),LTE
      LTE2=LTE
C-----
C LTR=65 ==> CALL PCB AIM
C LTR=66 ==> CALL PCB BCME RELEASE. ALSO ALLOWS INCREASED AIRSPEED.
C LTR=82 ==> CALL PCB PROGRAM TERMINATION
C LTR=83 ==> CALL TO BREAK OFF INPUT ROUTINE AND RESTART. MEANT TO
C             ALLOW CHANGING MIND WHILE IN MIDDLE OF PROGRAM.
C-----
      IF (LTR.EQ.65.CB.LTR2.EQ.65) CALL AIMFT
      IF (LTR.EQ.66.CB.LTR2.EQ.66) VMAX=VMAX2
      IF (LTR.EQ.82.CB.LTR2.EQ.82.AND.IMP.NE.0) GO TO 180
      IF (LTR.EQ.83.CB.LTR2.EQ.83) GO TO 140
      CALL GRAFIX
      ICT=ICT+1
      GO TO 130
C-----
C READ MILESTONES FROM THE TERMINAL

```



```

C-----
140  CCNTINUE
    CALL WIN (MINX,MAXX,18000.,12000.,0)
    CALL GSLT (1)
    CALL GBGARC (6000.,6000.,TARGX,TARGY,1.,1.,2.)
    CALL GSLT (0)
    CALL GSFRCE
    CALL XYPT
    CALL ZPT
C-----
C COMPUTE FLIGHT PARAMETERS AND CHECK FOR ERRORS OR USER COMMANDS
C-----
    IERR=0
    CALL EBRCEK (IERR)
    IF (IERR.EQ.0) GO TO 150
    WRITE (6,270)
    WRITE (6,280)
    READ (5,*) ICCB
    CALL ERTCMS ('CLRSCRN ')
    IF (ICCB.NE.1) GO TO 150
    CALL XYPT
C-----
C WAYPOINT ACCEPTED, FILL IN THE CURVE
C-----
150  CONTINUE
    CALL BIGCAL
    ICT=ICT+1
    IF (XOLD.NE.XNEW.OR.YOLD.NE.YNEW.OR.ZOLD.NE.ZNEW) GO TO 150
C-----
C LTR=65 ==> CALL FOR AIM
C LTR=66 ==> CALL FOR BOMB RELEASE. ALSO ALLOWS INCREASED AIRSPEED.
C LTR=82 ==> CALL FOR PROGRAM TERMINATION
C LTR=83 ==> CALL TO BREAK OFF INPUT ROUTINE AND RESTART. MEANT TO
C             ALLOW CHANGING MIND WHILE IN MIDDLE OF PROGRAM.
C-----
    IF (LTR.EQ.65.OR.LTR2.EQ.65) CALL AIMT
    IF (LTR.EQ.66.OR.LTR2.EQ.66) VMAX=VMAX2
    IF (LTR.EQ.82.OR.LTR2.EQ.82.AND.IMP.NE.0) GO TO 180
    IF (LTR.EQ.83.OR.LTR2.EQ.83) GO TO 160
    GO TO 140
C-----
C THIS SECTION RESETS THE DATA
C-----
160  CALL SCENE (1,ICNT)
    REWIND 19
C-----
C DRAW RETAINED MILESTONES
C-----
    REWIND 11
    ICT=ICNT
    DO 170 I=1,ICT
    READ (11,260) T(ICT),TX(ICT),WY(ICT),WZ(ICT),WU(ICT),WV(ICT),WW(IC
1T),WH(ICT),WP(ICT),WR(ICT),WG(ICT),LTR
    LTR2=LTR
    IF (LTR.EQ.65.OR.LTR2.EQ.65) CALL AIMT
    IF (LTR.EQ.66.OR.LTR2.EQ.66) VMAX=VMAX2
    IF (LTR.EQ.82.OR.LTR2.EQ.82.AND.IMP.NE.0) GO TO 180
    IF (LTR.EQ.83.OR.LTR2.EQ.83) GO TO 140
    CALL GRAFIX
170  CONTINUE
    GO TO 100
C-----
C THIS SECTION TERMINATES THE PROGRAM
C-----
180  CONTINUE
    LAST=IMP
    CALL ELFIN (LAST)
C-----
C CONVERT ANGLES FROM RADIANS TO DEGREES
C-----
    DO 190 I=1,ICT
    WH(I)=WH(I)*RTD
    WP(I)=WP(I)*RTD
    WR(I)=WR(I)*RTD
190  CONTINUE
C-----

```



```

C WRITE THE JCL CARDS
C -----
IF (IPNCH.LI.1) STCP
IF (IPNCH.EQ.2) GC TC 200
WRITE (6,290)
WRITE (6,300)
WRITE (6,310)
WRITE (6,320)
READ (5,400) INAM1(1),INAM1(2)
CALL FRTCMS ('CLSCFN')
200 IF (IPNCH.EQ.1) GC TC 210
WRITE (6,330)
WRITE (6,300)
WRITE (6,310)
WRITE (6,320)
READ (5,400) INAM2(1),INAM2(2)
CALL FRTCMS ('CLSCFN')
210 WRITE (6,340)
READ (5,*) IUNR
CALL FRTCMS ('CLSCFN')
WRITE (6,350)
READ (5,*) IANR
CALL FRTCMS ('CLSCFN')
C -----
C CHOOSE DESTINATION FOR OUTPUT: PRINTER OR READER FILE
C -----
220 WRITE (6,360)
WRITE (6,370)
READ (5,*) IDEST
IF ((IDEST.NE.1).AND.(IDEST.NE.2)) GO TO 220
CALL FRTCMS ('CLSCFN')
C -----
C PRESET LOADS MICE/POO1 DATA FILES CINTO USERS A1 DISK
C -----
CALL PRESET
C -----
C DUMMY INPUT ROUTINE TO LET USER KNOW WHERE S/HE IS IN THE PGM
C -----
WRITE (6,380)
READ (5,*) IGO
CALL FRTCMS ('CLSCFN')
C -----
C FORCE CALL TO CMS TO SUBMIT DATA FILES TO BATCH PROCESSING PGM
C -----
IF (IPNCH.EQ.2) GC TC 230
CALL FRTCMS ('EXEC','SUBMIT','POO1','DATA')
IF (IPNCH.EQ.1) GC TC 240
230 CALL FRTCMS ('EXEC','SUBMIT','MICE','DATA')
240 CONTINUE
C -----
C DUMMY INPUT ROUTINE TO LET USER KNOW PGM TERMINATION HAS BEEN REACHED
C -----
WRITE (6,390)
READ (5,*) IGC
STCP
250 FORMAT (3F10.2)
260 FCORMAT (11(F10.1,X),I3)
270 FCORMAT (3H )
280 FCORMAT (54H DC YOU WANT TO FIX THE ERROR:0=NO,USE THE POINT;1=YES)
290 FCORMAT (40H TO SUBMIT POO1 TO BATCH YOU MUST CREATE)
300 FCORMAT (31H A JOB CONTROL LANGUAGE "CARD".)
310 FCORMAT (51H ENTER A NAME -- IT MAY CONTAIN UP TO 8 CHARACTERS;)
320 FCORMAT (25H **DO NOT INSERT BLANKS**)
330 FCORMAT (44H TO SUBMIT **MICE** TO BATCH YOU MUST CREATE)
340 FCORMAT (46H ENTER YOUR USER ID NUMBER -- WITHOUT THE "P".)
350 FCORMAT (48H ENTER YOUR ACCT NUMBER -- PROVIDED BY THE PROF.)
360 FCORMAT (41H CHOOSE THE DESTINATION FOR YOUR OUTPUT: )
370 FCORMAT (3CH 1=PRINTER; 2=YOUR READER FILE)
380 FCORMAT (46H ENTER ANY **NUMBER** TO SUBMIT YOUR POO1/MICE,14H DATA
1 FILE(S).)
390 FCORMAT (4CH ENTER ANY **NUMBER** TO TERMINATE T2PIP)
400 FCORMAT (2A4)
END

```



```

C*****
C SUBROUTINE BEGIN      THIS SUBROUTINE INITIALIZES THE GRAPHICS AND REQUESTS
C                        THE USER OPTIONS.
C*****
      SUBROUTINE BEGIN
      COMMON /OPT/ IGUN,IFNCH,IEXT,ISAM,IMP,KON,IMPZ
C-----
C READ USER OPTIONS
C-----
      CALL FRTCMS ('CLRSCN ')
10    WRITE (6,40)
      READ (5,*) IGUN
      IF ((IGUN.NE.0).AND.(IGUN.NE.1).AND.(IGUN.NE.2)) GO TO 10
      CALL FRTCMS ('CLRSCN ')
      IF (IGUN.NE.1) GO TO 20
      WRITE (6,50)
      WRITE (6,60)
      READ (5,*) IMPZ
      CALL FRTCMS ('CLRSCN ')
20    WRITE (6,70)
      READ (5,*) IMP
      IF ((IMP.NE.0).AND.(IMP.NE.1)) GO TO 20
      CALL FRTCMS ('CLRSCN ')
30    WRITE (6,80)
      READ (5,*) KON
      IF ((KON.NE.1).AND.(KON.NE.2)) GO TO 30
      CALL FRTCMS ('CLRSCN ')
      IF (KON.NE.1) CALL CONCHG
C-----
C INITIALIZE THE GRAPHICS AND CLEAR SCREEN
C-----
      CALL DSINIT
      CALL GSERSE
      RETURN
40    FORMAT (50H GUNS:0=DISK FILE;1=TERMINAL;2=PRESET(GOUGE SET OF GUNS
1)
50    FORMAT (55H ** CHOICE 1 NEGATES HAVING TO GO OVER TO THE ALTIMETER
1)
60    FORMAT (55H GUN/SAM SITE ALTITUDE:1=FIXED TO 20M;2=USER DEFINEABLE
1)
70    FORMAT (50H AIRCRAFT MILESTONE INPUT: 0=DISK FILE; 1=TERMINAL)
80    FORMAT (50H FLIGHT & GAME PARAMETERS: 1=DEFAULT; 2=USER INPUT)
90    END

```



```

C *****
C SUBROUTINE CONCHG
C      THIS SUBROUTINE ALLOWS OPERATOR CHANGES TO THE
C      SIMULATION PARAMETERS.
C *****
      SUBROUTINE CONCHG
      COMMON /AED2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTU,DELTZ
      COMMON /AEE4/ THMAX,ANMAX,CLMAX,GEE,AKK,RHOSSL,AL,CDO,CDK,AREA
      COMMON /PAR4/ APPMAX,HTMIN,HTMAX,POPMIN,BAALT,BRAIN,BRMAX,BRMIN,B
      IRRMAX,RTD,AMTF
      COMMON /TAR/ TARGX,TARGY
C -----
C WRITE INSTRUCTIONS
C -----
10  WRITE (6,190)
    WRITE (6,200)
    WRITE (6,210)
    WRITE (6,220)
    WRITE (6,230)
    WRITE (6,240)
    WRITE (6,250)
    WRITE (6,260)
    WRITE (6,270)
    READ (5,*) K
    IF ((K.LE.0).OR.(K.GT.17)) GO TO 10
    GO TO (180,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170
1) , K
20  WRITE (6,280)
    READ (5,*) CLMAX
    GO TO 10
30  WRITE (6,290)
    READ (5,*) AL
    GO TO 10
40  WRITE (6,300)
    READ (5,*) VSTALL
    GO TO 10
50  WRITE (6,310)
    READ (5,*) ANMAX
    GO TO 10
60  WRITE (6,320)
    READ (5,*) HTMIN
    GO TO 10
70  WRITE (6,330)
    READ (5,*) HTMAX
    GO TO 10
80  WRITE (6,340)
    READ (5,*) POPMIN
    GO TO 10
90  WRITE (6,350)
    READ (5,*) APPMAX
    GO TO 10
100 WRITE (6,360)
    READ (5,*) THMAX
    GO TO 10
110 WRITE (6,370)
    READ (5,*) CDO
    GO TO 10
120 WRITE (6,380)
    READ (5,*) CDK
    GO TO 10
130 WRITE (6,390)
    READ (5,*) VMAX1
    GO TO 10
140 WRITE (6,400)
    READ (5,*) VMAX2
    GO TO 10
150 WRITE (6,410)
    READ (5,*) TARGX,TARGY
    GO TO 10
C -----
C READ NEW PARAMETER VALUES FROM THE DISK
C -----
160 REWIND 16
    READ (16,450) CLMAX,AL,VSTALL,ANMAX,HTMIN,HTMAX
    READ (16,450) POPMIN,APPMAX,THMAX,CDO,CDK,VMAX1
    READ (16,450) VMAX2,TARGX,TARGY

```



```

      GO TO 10
C-----
C WRITE PARAMETER VALUES TO THE TERMINAL
C-----
170  WRITE (6,420)
      WRITE (6,450) CLMAX,WL,VSTALL,ANMAX,HTMIN,HTMAX
      WRITE (6,430)
      WRITE (6,450) POPMIN,APPMAX,THMAX,CDO,CDK,VMAX1
      WRITE (6,440)
      WRITE (6,450) VMAX2,TARGX,TARGY
      GO TO 10
C-----
C SAVE PARAMETER VALUES ON DISK AND RETURN TO CALLING ROUTINE
C-----
180  REWIND 16
      WRITE (16,450) CLMAX,WL,SPDMIN,GMAX,HTMIN,HTMAX
      WRITE (16,450) PCEMIN,APPMAX,THMAX,CDO,CDK,VMAX1
      WRITE (16,450) VMAX2,TARGX,TARGY
      RETURN
190  FORMAT (25H )
200  FORMAT (5CH SELECT PARAMETER TO BE CHANGED: 1=NO MORE CHANGES)
210  FORMAT (48H 2=MAX LIFT COEFF; 3=WING LOADING; 4=STALL SPEED)
220  FORMAT (47H 5=MAX G FORCE; 6=MIN ALT; 7=MAX ALT(<2200 MTR))
230  FORMAT (42H 8=DISTANCE TO TARGET BEFORE POPUP ALLOWED)
240  FORMAT (63H 9=MAX APPROACH ALT; 10=MAX I/W RATIO; 11=DRAG COEF. W/
10  LIFT)
250  FORMAT (44H 12=LIFT DRAG CONSTANT; 13=MAX SPD WITH BOMB)
260  FORMAT (44H 14=MAX SPD W/C BOMB; 15=TARGET COORDINATES )
270  FORMAT (43H 16=READ PARAMETER FILE; 17=LIST PARAMETERS)
280  FORMAT (27H ENTER MAX LIFT COEFFICIENT)
290  FORMAT (19H ENTER WING LOADING)
300  FORMAT (18H ENTER STALL SPEED)
310  FORMAT (26H ENTER MAX G FORCE ALLOWED)
320  FORMAT (23H ENTER MINIMUM ALTITUDE)
330  FORMAT (47H ENTER MAXIMUM ALTITUDE (LESS THAN 2000 METERS))
340  FORMAT (55H ENTER MINIMUM DISTANCE TO TARGET BEFORE POP-UP ALLOWED)
1)
350  FORMAT (32H ENTER MAXIMUM APPROACH ALTITUDE)
360  FORMAT (35H ENTER MAX THRUST TO HEIGHT ALLOWED)
370  FORMAT (37H ENTER DRAG COEFFICIENT FOR ZERO LIFT)
380  FORMAT (25H ENTER LIFT DRAG CONSTANT)
390  FORMAT (32H ENTER MAX SPEED CARRYING A BOMB)
400  FORMAT (39H ENTER MAX SPEED AFTER BOMB IS RELEASED)
410  FORMAT (33H ENTER TARGET X AND Y COORDINATES)
420  FORMAT (55H MAX CL WING LD STALL SPD MAX G MIN.HT. MAX.HT.
1)
430  FORMAT (75H POP-UP DIST APPR.HT. MAX THRUST CDO DRAG CONSTANT
1 MAX SPD WITH BOMB)
440  FORMAT (55H MAX SPD W/O BOMB TARGET X COORD. TARGET Y COORD.
1)
450  FORMAT (6F12.4)
      END

```



```

C *****
C SUBROUTINE SCENE
C THIS ROUTINE DRAWS THE ATTACK MAP AND ASSOCIATED
C DISPLAYS. DURING RESET IT OBTAINS THE LAST MILE-
C STONE THE USER WISHES TO RETAIN.
C *****
C SUBROUTINE SCENE (IRC,ICNT)
C COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C COMMON /MN/ TBAUD
C COMMON /ABC1/ AX(1000),AY(1000),AZ(1000),AU(1000),AV(1000),AW(1000)
C 1,WH(1000),WP(1000),WR(1000),WG(1000),I(1000),ICT
C COMMON /TAR/ TARGX,TARGY
C DIMENSION VEL(200)
C VEL(ICNT)=SQRT(WU(ICNT)**2+WV(ICNT)**2+WW(ICNT)**2)
C REAL*4 GX,GY
C LOGICAL *1CHARS(20)
C ICNT=0
C -----
C ERASE THE TEK618 SCREEN
C -----
C CALL GSERSE
C CALL DSTERA
C -----
C READ USER OPTIONS:BCRIER AND GUN/MISS POSITS OR FILL IN MAP ALSO.
C -----
C WRITE (6,100)
C READ (5,*) MAP
C IF (IRC.EQ.0) GO TO 10
C WRITE (6,90)
C READ (5,*) ICNT
C CALL PRTCMS ('CLRSCRN ')
C -----
C DRAW THE X,Y MAP
C THE ROUTINE READS X,Y PAIRS FROM FILE 09. A -2.0,0 INDICATES
C END OF FILE, -0.5,0 INDICATES A MOVE,ALL OTHER VALUES ARE PO-
C SITIVE AND RESULT IN A DRAW TO THAT POINT.
C -----
C RE-INITIALIZE THE TEK618 SCREEN
C -----
10 CALL DSINIT
C CALL WIN (MINX,MAXX,18000.,12000.,1)
C IF (MAP.EQ.0) GO TO 40
C REWIND 9
20 READ (9,80) GX,GY
C READ (9,80) GX,GY
C CALL GBVECT (0.,GX,GY)
30 READ (9,80) GX,GY
C IF (GX.LT.-1) GO TO 40
C IF (GX.LT.0) GO TO 20
C CALL GBVECT (1.,GX,GY)
C GO TO 30
C -----
C MARK THE TARGET: GUNLOC PUTS A + AT GUN/MISS POSIT AND CIRCUMSCRIBES
C -----
40 CALL GUNLOC (TARGX,TARGY,150.)
C CALL GUNLOC (TARGX,TARGY,300.)
C -----
C DRAW TIC MARKS EVERY 2000 MTRS
C -----
C DO 50 I=1,8
C IW=2*I
C -----
C THIS BLOCK DRAWS THE X-AXIS TIC MARKS
C -----
C CALL GBVECT (0.,I*2000.,1000.)
C CALL GBVECT (1.,I*2000.,0.)
C CALL GAXITC (1,IW,2,CHARS)
C CALL GBCHAR (I*2000.,0.,2.,0.,2,CHARS)
C -----
C THIS BLOCK DRAWS THE Y-AXIS TIC MARKS
C -----
C IF (I.GE.6) GO TO 50
C CALL GBVECT (0.,1000.,I*2000.)
C CALL GBVECT (1.,0.,2000.*I)
C CALL GAXITC (1,IW,2,CHARS)
C CALL GBCHAR (-60.,I*2000.,2.,0.,2,CHARS)

```



```

50  CONTINUE
    I1=MINY
    I2=MAXY
    MAXY=MINY-15
    MINY=MINY-200
    CALL WIN (MINX+275,MINX+2200,195.,185.,0)
    CALL GBCHAR (10.,110.,2.5,0.,30,'KEY MARKS IN 2000 METER INTVLS')
    CALL GBCHAR (10.,40.,2.5,0.,25,'ZERO DEGREES ALONG X AXIS')
    CALL GSFCE
    MAXY=I2
    MINY=I1
C-----
C  DRAW THE ALTIMETER
C-----
    CALL WIN (MINX1,MAXX1,350.,2200.,1)
    DO 50 I=2,20,2
    CALL GBVECT (0.,0.,I*100.)
    CALL GBVECT (1.,350.,I*100.)
    CALL GAXITC (1,1,2,CHARS)
    CALL GBCHAR (0.,I*100.,2.,0.,2,CHARS)
60  CONTINUE
    DO 70 I=1,6
    CALL GBVECT (0.,I*50.,100.)
    CALL GBVECT (1.,I*50.,0.)
70  CONTINUE
C-----
C  MARK RESET POINT VELOCITY AND ALTITUDE AS OPERATOR AID
C-----
    IF (IRO.NE.0 .AND. ICNT.GT.1) CALL GBGDOT (9.,VEL(ICNT),#Z(ICNT))
    MAXY=MINY-15
    MINY=MINY-200
    CALL WIN (MINX+2800,MINX+3900,195.,185.,0)
    CALL GBCHAR (10.,110.,2.0,0.,29,'ALT MARKS IN 200 METER INTVLS')
    CALL GBCHAR (10.,60.,2.0,0.,33,'VELO IN 50 1/3 INTVL ALONG X AXIS'
1)
    CALL GSFCE
    MAXY=I2
    MINY=I1
    RETURN
80  FORMAT (2F10.2)
90  FORMAT (38H AT WHICH POINT DO YOU WANT TO RESTART)
100 FORMAT (38H VILLAGE: 0=NOT DRAWN; 1=VILLAGE DRAWN)
    END

```



```

C *****
C SUBROUTINE WIN      THIS ROUTINE DEFINES A WINDOW EXTENDING FROM LX TO MX
C                     ON THE HORIZONTAL AXIS, AND FROM MINY TO MAXY ON THE
C                     VERTICAL AXIS. THE HORIZONTAL RANGE IS RX, THE
C                     VERTICAL RANGE IS RY.
C *****
      SUBROUTINE WIN (LX,MX,RX,RY,JMP)
      COMMON /MN/ IBAUD
      COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
      REAL*4 WAN1(4)
      INTEGER*4 V1(4)
      DATA WAN1(1)/0./,WAN1(2)/0./
C-----
C DEFINE THE WINDOW
C-----
      V1(2)=MINY
      V1(4)=MAXY
      WAN1(3)=RX
      WAN1(4)=RY
      V1(1)=LX
      V1(3)=MX
      CALL GASVIE (V1,J)
      CALL GASWIN (WAN1,0)
      IF (JMP .EQ. 0) GC TC 20
C-----
C FLASH THE DEFINED WINDOW
C-----
      ICNT=4+5C*IBAUD
      DO 10 I=1,ICNT
      CALL GBVECT (0.,0.,0.)
      CALL GBVECT (1.,WAN1(1),WAN1(4))
      CALL GBVECT (1.,WAN1(3),WAN1(4))
      CALL GBVECT (1.,WAN1(3),WAN1(2))
      CALL GBVECT (1.,WAN1(1),WAN1(2))
C-----
C YOU DO NOT WANT A "CALL GSFRCE" HERE. WINDOW DEFINES THE GRAPHICS
C PCET OR "WINDOW" YOU ARE "OPENING". THE OTHER SUBROUTINES ARE PASSING
C THINGS THROUGH THE OPEN WINDOW. "CALL GSFRCE" CLOSSES THE WINDOW.
C-----
10  CONTINUE
20  RETURN
      END

```



```

C*****
C SUBROUTINE GUNLOC
C      THIS SUBROUTINE DRAWS EFFECTIVE FIRE RADIUS CIRCLES
C      AROUND THE DEFENSIVE AAA BATTERIES.
C*****
C      SUBROUTINE GUNLOC (GX,GY,RAD)
C      COMMON /MN/ IEAUC
C      COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C      REAL*4 X,Y,DX,DY
C      X=GX
C      Y=GY
C-----
C IDENTIFY THE MAP AND PLACE A '+' AT THE LOCATION
C-----
C      CALL WIN (MINX,MAXX,18000.,12000.,0)
C      CALL GBVECT (0.,X-75.,Y)
C      CALL GBVECT (1.,X+75.,Y)
C      CALL GBVECT (0.,X,Y+75.)
C      CALL GBVECT (1.,X,Y-75.)
C-----
C DRAW A CIRCLE AROUND THE SITE
C-----
C      CALL GSLT (14)
C      CALL GEGARC (RAD,RAD,X,Y,1.,1.,2.)
C      CALL GSLT (0)
C      CALL GSFRCE
C      RETURN
C      END

C*****
C SUBROUTINE XYPT
C      THIS SUBROUTINE OBTAINS THE X AND Y COORDINATES
C      AND COMMAND FOR EACH DESIGNATED POINT (MILESTONE).
C*****
C      SUBROUTINE XYPT
C      COMMON /LCC/ X1,Z1,V1,ITR,ITR2
C      COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C      CALL BELL
C-----
C FLASH MAP AND GET X,Y AND THE FIRST COMMAND
C-----
C      CALL WIN (MINX,MAXX,18000.,12000.,1)
C      CALL SPOT (X1,Y1,ITR)
C      CALL GSFRCE
C      CALL BELL
C      RETURN
C      END

C*****
C SUBROUTINE ZVPT
C      THIS SUBROUTINE OBTAINS THE ALTITUDE AND, DEPENDING
C      ON THE CIRCUMSTANCES, THE VELOCITY OF THE DESIGNATED
C      PCINT.
C*****
C      SUBROUTINE ZVPT
C      COMMON /LCC/ X1,X1,Z1,V1,ITR,ITR2
C      COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C-----
C FLASH ALTIMETER AND GET VEL,ALT AND THE SECOND COMMAND
C-----
C      CALL WIN (MINX1,MAXX1,350.,2200.,1)
C      CALL SPOT (V1,Z1,ITR2)
C      CALL GSFRCE
C      RETURN
C      END

```



```

C*****
C SUBROUTINE SPCT      THIS RETURNS AN X-Y PAIR AND A COMMAND VALUE L1
C                      THE ROUTINE ALSO CREATES A DOT AT POINT X,Y.
C*****
C SUBROUTINE SPOT (X,Y,L1)
C   REAL*4 TX,TY,DCT
C   INTEGER*4 L,M,CUT,KEY
C-----
C READ THE CURSOR AND PICK UP USER COMMAND
C-----
10  CALL GBRXYC (M,L,KEY,TX,TY)
    CALL GAXCTI (1,KEY,1,LEN,OUT)
    DOT=40.
C-----
C MARK THE SPOT
C-----
    CALL GBGECT (DCT,IX,IY)
    X=TX
    Y=TY
C-----
C CONVERT THE COMMAND TO THE PROPER FORMAT
C-----
    L1=0
    IF (OUT.EQ.1) L1=65
    IF (OUT.EQ.2) L1=66
    IF (OUT.EQ.3) L1=82
    IF (OUT.EQ.4) L1=83
    RETURN
END

C*****
C SUBROUTINE GUNCHK  THIS SUBROUTINE CHECKS GUN LOCATIONS AGAINST
C                      EMPLACEMENT RULES.
C*****
C SUBROUTINE GUNCHK (X,Y,IERR)
C   COMMON /TAR/ TARGX,TARGY
C-----
C COMPUTE DISTANCE TO TARGET
C-----
    DIST=SQRT((X-TARGX)**2 + (Y-TARGY)**2)
    IERR=0
    IF (DIST.GE. 3000.) RETURN
    IERR=13
    CALL ERRMK (IERR)
    RETURN
END

C*****
C SUBROUTINE SAMCHK  THIS SUBROUTINE CHECKS THE MISSILE EMPLACEMENT
C                      AGAINST GAME LIMITATIONS.
C*****
C SUBROUTINE SAMCHK (X,IERR)
C   IERR=0
C   IF (X.GE. 6000.) RETURN
C   IERR=19
C   CALL ERRMK (IERR)
C   RETURN
END

```



```

C*****
C SUBROUTINE ERMK
C      THIS SUBROUTINE SENDS PERMETS AND ERROR MESSAGES
C      TO THE USER.
C*****
C SUBROUTINE ERMK (IERR)
COMMON /HM/ TBAUD
COMMON /AEC1/ WX(1000), WY(1000), WZ(1000), WU(1000), WV(1000), WW(1000)
1) ,WH(1000), WP(1000), WR(1000), WG(1000), T(1000), ICT
COMMON /MNI/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C-----
C FLASH THE PERMPT WINDOW
C-----
      CALL WIN (MINX2,MAXX2,2700.,185.,1)
      CALL GSFRCE
      GO TO (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
180), IERR
10      WRITE (6,200)
      GC TO 190
20      WRITE (6,210)
      GC TO 190
30      WRITE (6,220)
      GC TO 190
40      WRITE (6,230)
      GC TO 190
50      WRITE (6,240)
      GC TO 190
60      WRITE (6,250)
      GC TO 190
70      WRITE (6,260)
      GC TO 190
80      WRITE (6,270)
      GC TO 190
90      WRITE (6,280)
      GC TO 190
100     WRITE (6,290)
      GC TO 190
110     WRITE (6,300)
      GC TO 190
120     WRITE (6,310)
      GC TO 190
130     WRITE (6,320)
      GC TO 190
140     WRITE (6,330)
      GC TO 190
150     WRITE (6,340)
      GC TO 190
160     WRITE (6,350)
      GC TO 190
170     WRITE (6,360)
      GC TO 190
180     WRITE (6,370)
      GC TO 190
190     RETURN
200     FORMAT (25H MAXIMUM BRAKING EXCEEDED)
210     FORMAT (15H MAX G EXCEEDED)
220     FORMAT (12H ALT TOO LOW)
230     FORMAT (13H ALT TOO HIGH)
240     FORMAT (6H STALL)
250     FORMAT (15H PCF-UE TOO LOW)
260     FORMAT (13H BME DECP LOW)
270     FORMAT (12H BSB DECP HI)
280     FORMAT (15H TCC FAR FM TGT)
290     FORMAT (17H HDG>5 DEG TO TGT)
300     FORMAT (19H FINAL RUN<2.33 SEC)
310     FORMAT (21H NO HCBIZCNTAL MOTION)
320     FORMAT (17H TOC CLOS TO TGT)
330     FORMAT (23H ENTER MISSILE LOCATION)
340     FORMAT (26H ENTER AIRCRAFT MILESTONES)
350     FORMAT (20H ENTER GUN LOCATIONS)
360     FORMAT (20H MAX THRUST EXCEEDED)
370     FORMAT (28H X COORDINATE LESS THAN 6000)
      END

```



```

C*****
C SUBROUTINE BIGCAL      THIS SUBROUTINE OPERATES AT A LEVEL BELOW MAIN
C                        PROGRAM AND ACTS AS A DRIVER FOR THE SUBROUTINES
C                        LISTED WITHIN.
C*****
SUBROUTINE BIGCAL
COMMON /AEC1/ WX(1000), WY(1000), WZ(1000), WU(1000), WV(1000), WW(1000
1), WH(1000), WP(1000), WR(1000), WG(1000), T(1000), ICT
COMMON /AEC2/ XNEW, DELX, XTEMP, XOLD, YNEW, DELY, YTEMP, YOLD, ZNEW, DELZ,
1ZTEMP, ZOLD
COMMON /AEC3/ VELCN, VELOT, VELCC, N, UNEW, UTEMP, UOLD, VNEW, VTEMP, VOLD,
1VNEW, VTEMP, VOLD
COMMON /AEC4/ AG1, AH1, AI1, AJ1, AJ2
COMMON /AEC5/ VSTALL, VMAX, VMAX1, VMAX2, DELTEE, DELTV, DELTE
COMMON /AEE5/ FWDACC, AN, MCONT
N=0
MCONT=0
CALL TENFERM
CALL NEWCLD
CALL VELCS
CALL MVDCTS
CALL DELVEC
CALL IMPENT
CALL IMPVEL
CALL INRTEM
CALL ANGLES
CALL ROTRAT
CALL ACTERM1
CALL ACTERM2
CALL ACTERM3
CALL ACRVEL
CALL ACACC
CALL ACLM7S
CALL ACHANGE
CALL AYENT
CALL FILE
ICT=ICT+1
RETURN
END

```



```

C*****
C SUBROUTINE TRNFRM TRANSFCRM TAKES THE REAL VALUE VARIABLES SUPPLIED
C BY THE GRAPHICS SUBROUTINES AND LOADS THEM INTO
C VARIABLES SPECIFIC TO THIS AIRCRAFT TRAJECTORY
C GENERATING PACKAGE.
C*****
SUBROUTINE TRNFRM
COMMON /LCC/ X1,Y1,Z1,V1,ITR,LIR2
COMMON /ABC2/ ANEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1WNEW,WTEMP,WOLD
COMMON /ABES/ FWDACC,AN,MCONT
XNEW=X1
YNEW=Y1
ZNEW=Z1
VELCN=V1
RETURN
END

```

```

C*****
C SUBROUTINE NEWOLD NEWOLD GENERATES LENGTH VECTOR COMPONENTS BETWEEN
C THE CURRENT WAYPOINT (OLD) AND THE DESIRED WAYPOINT
C USER HAS JUST ENTERED (NEW).
C*****
SUBROUTINE NEWOLD
COMMON /ABC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /ABC4/ AG1,AH1,AI1,AJ1,AJ2
COMMON /ABES/ FWDACC,AN,MCONT
AG1=XNEW-XOLD
AH1=YNEW-YOLD
AI1=ZNEW-ZOLD
AJ1=SQRT(AG1**2+AH1**2+AI1**2)
AJ2=SQRT(AG1**2+AH1**2)
RETURN
END

```



```

C *****
C SUBROUTINE MVELOS
C MVELCS (MAP VELOCITY COMPONENTS) DETERMINES THE
C VELOCITY COMPONENTS BETWEEN CURRENT WAYPOINT (OLD)
C AND THE JUST ENTERED WAYPOINT (NEW). THE SUBROUTINE
C CONTAINS A FLAG (N) TO AVOID RESETTING THE VELOCITY
C COMPONENTS UNTIL EITHER A NEW WAYPOINT IS RECEIVED
C OR AN AIRCRAFT LIMITATION WHICH IS FATAL IS REACHED
C IN THE SUBSEQUENT ITERATIVE SUBROUTINES. THE
C COORDINATE SYSTEM IS TERMED MAP, I.E., Y POINTS NORTH
C *****

```

```

SUBROUTINE MVELOS
COMMON /AEC4/ AG1,AH1,AI1,AJ1,AJ2
COMMON /AEC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1 WNEW,WTEMP,WOLD
COMMON /AEE5/ FWDACC,AN,MCONT
M=N
IF (N.GT.0) GO TO 10
UNEW=VELCN*(AG1/AJ1)
VNEW=VELCN*(AH1/AJ1)
WNEW=VELCN*(AI1/AJ1)
10 CONTINUE
N=1
RETURN
END

```

```

C *****
C SUBROUTINE MVDOTS
C MVDOTS (MAP VELOCITY DOT COMPONENTS) COMPUTES THE
C LINEAR ACCELERATION COMPONENTS
C IN MAP COORDINATES. (Y POINTS NORTH)
C *****

```

```

SUBROUTINE MVDOTS
COMMON /AEC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1 ZTEMP,ZOLD
COMMON /AEC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1 WNEW,WTEMP,WOLD
COMMON /AEC4/ AG1,AH1,AI1,AJ1,AJ2
COMMON /AEE1/ UDOT,VDOT,WDOT
COMMON /AEE5/ FWDACC,AN,MCONT
ACC=(VELCN**2-VELCC**2)/(2*AJ1)
UDOT=ACC*(AG1/AJ1)
VDOT=ACC*(AH1/AJ1)
WDOT=ACC*(AI1/AJ1)
RETURN
END

```



```

C*****
C SUBROUTINE DELVEC
C      DELVEC (DELTA VECTORS) UTILIZES STANDARD DYNAMIC
C      EQUATIONS TO GENERATE LINEAR TRANSLATION INCREMENTS.
C      IT FURTHER LOOKS TO SEE WHETHER THESE INCREMENTS
C      EXCEED THE DISTANCE LEFT TO GO TO THE NEW WAYPOINT
C*****
      SUBROUTINE DELVEC
      COMMON /ABC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
      COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1WNEW,WTEMP,WOLD
      COMMON /ABC4/ AG1,AH1,AI1,AJ1,AJ2
      COMMON /ABD2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTV,DELTAZ
      COMMON /ABE1/ UDOT,VDOT,WDOT
      COMMON /ABE5/ FWCACC,AN,MCONT
      DELX=UOLD*DELTEE+.5*UDOT*DELTEE**2
      DELY=VOLD*DELTEE+.5*VDOT*DELTEE**2
      DELZ=WOLD*DELTEE+.5*WDOT*DELTEE**2
      IF (ABS(DELX).GT.ABS(AG1)) DELX=AG1
      IF (ABS(DELY).GT.ABS(AH1)) DELY=AH1
      IF (ABS(DELZ).GT.ABS(AI1)) DELZ=AI1
      RETURN
      END

```

```

C*****
C SUBROUTINE TMPPNT
C      TMPENT (TEMPORARY POINT) GENERATES A TEMPORARY
C      POSITION IN SPACE BASED UPON THE DELTA VECTORS
C*****
      SUBROUTINE TMPENT
      COMMON /ABC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
      COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1WNEW,WTEMP,WOLD
      COMMON /ABD2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTV,DELTAZ
      COMMON /ABE5/ FWCACC,AN,MCONT
      XTEMP=XOLD+DELX
      YTEMP=YOLD+DELY
      ZTEMP=ZOLD+DELZ
      IF (DELX.EQ.AG1) XTEMP=XNEW
      IF (DELY.EQ.AH1) YTEMP=YNEW
      IF (DELZ.EQ.AI1) ZTEMP=ZNEW
      RETURN
      END

```



```

C*****
C SUBROUTINE TMPVEL
C*****
C TMPVEL (TEMPORARY VELOCITY COMPONENTS) GENERATES
C THE VELOCITY COMPONENTS ASSOCIATED WITH THE TEMPORARY
C POINT IN SPACE (MAP COORDINATES).
C*****
C SUBROUTINE TMPVEL
COMMON /AEC2/ XNEW, DELX, XTEMP, XOLD, YNEW, DELY, YTEMP, YOLD, ZNEW, DELZ,
1 ZTEMP, ZOLD
COMMON /AEC3/ VELCN, VELOT, VELOC, N, UNEW, UTEMP, UOLD, VNEW, VTEMP, VOLD,
1 WNEW, WTEMP, WOLD
COMMON /AEC4/ AG1, AH1, AI1, AJ1, AJ2
COMMON /ABD2/ VSTALL, VMAX, VMAX1, VMAX2, DELTEE, DELTV, DELTAZ
COMMON /AEE1/ UDOT, VDOT, WDOT
COMMON /AEE5/ FWDACC, AN, MCONT
UTEMP=UOLD+UDOT*DELTEE
VTEMP=VOLD+VDOT*DELTEE
WTEMP=WOLD+WDOT*DELTEE
IF (ABS(UTEMP).GT.ABS(UNEW).AND.MCONT.EQ.1) UTEMP=UNEW
IF (ABS(VTEMP).GT.ABS(VNEW).AND.MCONT.EQ.1) VTEMP=VNEW
IF (ABS(WTEMP).GT.ABS(WNEW).AND.MCONT.EQ.1) WTEMP=WNEW
IF (DELX.EQ.AG1) UTEMP=UNEW
IF (DELY.EQ.AH1) VTEMP=VNEW
IF (DELZ.EQ.AI1) WTEMP=WNEW
IF (DELX.NE.0.) GO TO 10
UDOT=0.
GO TO 20
CONTINUE
10 IF (UTEMP.EQ.UNEW) UDOT=(UTEMP**2-UOLD**2)/(2*DELX)
20 CONTINUE
IF (DELY.NE.0.) GO TO 30
VDOT=0.
GO TO 40
CONTINUE
30 IF (VTEMP.EQ.VNEW) VDOT=(VTEMP**2-VOLD**2)/(2*DELY)
40 CONTINUE
IF (DELZ.NE.0.) GO TO 50
WDOT=0.
GO TO 60
CONTINUE
50 IF (WTEMP.EQ.WNEW) WDOT=(WTEMP**2-WOLD**2)/(2*DELZ)
60 CONTINUE
VELOT=SQRT(UTEMP**2+VTEMP**2+WTEMP**2)
RETURN
END

```

```

C*****
C SUBROUTINE INTRM
C*****
C INTRM (INITIAL ROTATIONAL TRANSFORMATION) TRANSFORMS
C MAP COORDINATE COMPONENTS TO A SYSTEM USED BY FLIGHT
C DYNAMICISTS (Y POINTS SOUTH AND Z POINTS DOWN)
C*****
C SUBROUTINE INTRM
COMMON /ABC3/ VELCN, VELOT, VELOC, N, UNEW, UTEMP, UOLD, VNEW, VTEMP, VOLD,
1 WNEW, WTEMP, WOLD
COMMON /AEE1/ UDOT, VDOT, WDOT
COMMON /AEE5/ FWDACC, AN, MCONT
COMMON /ABF1/ G1, V1, A1, U2, V2, W2, U3, V3, W3
COMMON /ABF2/ UD1, VD1, WD1, UD2, VD2, WD2, UD3, VD3, WD3
U1=UTEMP
V1=-VTEMP
W1=-WTEMP
UD1=UDOT
VD1=-VDOT
WD1=-WDOT
RETURN
END

```



```

C*****
C SUBROUTINE ANGLES
C*****
C ANGLS COMPUTES THE ANGLE CHANGE INCREMENTS CAUSED
C BY MOVEMENT FORWARD TO THE TEMPORARY POINT. PSINew
C IS THE HEADING CHANGE, REFERENCED TO THE X AXIS.
C THENew IS THE PITCH CHANGE REFERENCED TO THE
C HORIZONTAL PLANE. PHINew IS THE ROLL INCREMENT ABOUT
C THE X AXIS.
C*****
SUBROUTINE ANGLES
COMMON /AEC2/ XNEW,DELX,XTEMP,XCLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /AEC3/ VELCN,VELOT,VELOO,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1VNEW,VTEMP,VCLD
COMMON /AEC4/ AG1,AH1,AI1,AJ1,AJ2
COMMON /AED1/ AA1,AB1,AC1
COMMON /AED2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTV,DELTZ
COMMON /AED4/ PSINew,THENew,PHINew,PSICLD,THEOLD,PHIOLD
COMMON /AEE5/ FWIACC,AN,MCONT
DS=SQRT(DELX**2+DELY**2+DELZ**2)
IF (DELX.NE.0.) GC TO 10
IF (DELY.EQ.0..AND.DELZ.LT.0.) PSINew=AA1
IF (DELX.EQ.0..AND.DELZ.GT.0.) PSINew=-AA1
GC TO 20
10 CONTINUE
PSINew=-ATAN(DELY/DELX)
IF (DELX.GT.0..AND.DELZ.EQ.0.) PSINew=0.
IF (DELX.LT.0..AND.DELZ.LT.0.) PSINew=AB1+PSINew
IF (DELX.LT.0..AND.DELZ.EQ.0.) PSINew=AB1
IF (DELX.LT.0..AND.DELZ.GT.0.) PSINew=PSINew-AB1
20 CONTINUE
THENew=ASIN(DELZ/DS)
TRNRAT=(PSINew-PSICLD)/DELTEE
IF (PSINew.GT.0.) PHINew=ATAN(ABS(TRNRAT*VELOT/9.7953))
IF (PSINew.LT.0.) PHINew=-ATAN(ABS(TRNRAT*VELOT/9.7953))
IF (PSINew.EQ.PSICLD) PHINew=PHIOLD
RETURN
END

```

```

C*****
C SUBROUTINE ROTRAT
C*****
C ROTRAT (ROTATIONAL RATES) COMPUTES THE ANGLE DOT
C COMPONENTS IN THE INERTIAL REFERENCE SYS. (THE ONE
C WITH Y POINTING SOUTH AND Z DOWN).
C*****
SUBROUTINE ROTRAT
COMMON /AEC2/ XNEW,DELX,XTEMP,XCLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /AED2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTV,DELTZ
COMMON /AED3/ PSIDOT,THEDOT,PHIDOT
COMMON /AED4/ PSINew,THENew,PHINew,PSICLD,THEOLD,PHIOLD
COMMON /AEE5/ FWIACC,AN,MCONT
PSIDOT=(PSINew-PSICLD)/DELTEE
THEDOT=(THENew-THEOLD)/DELTEE
PHIDOT=(PHINew-PHIOLD)/DELTEE
RETURN
END

```



```

C*****
C SUBROUTINE ACTRM1
C      ACTRM1 (AIRCRAFT TRANSFORMATION MATRIX #1) EFFECTS
C      ROTATION OF AIRCRAFT ABOUT THE Z1 AXIS (POS 2
C      POINTS DOWN TO EARTH).
C*****
C SUBROUTINE ACTRM1
COMMON /ABD4/ PSINEX,THENEW,PHINEX,PSICLD,THEOLD,PHIOLD
COMMON /AEE5/ FWDACC,AN,MCONT
COMMON /ABF1/ U1,V1,W1,U2,V2,W2,U3,V3,W3
COMMON /AEF2/ UD1,VD1,WD1,UD2,VD2,WD2,UD3,VD3,WD3
ANG=PSINEX
U2=COS(ANG)*U1+SIN(ANG)*V1
V2=-SIN(ANG)*U1+CCS(ANG)*V1
W2=W1
UD2=COS(ANG)*UD1+SIN(ANG)*VD1
VD2=-SIN(ANG)*UD1+CCS(ANG)*VD1
WD2=WD1
RETURN
END

```

```

C*****
C SUBROUTINE ACTRM2
C      ACTRM2 ROTATES THE AIRCRAFT ABOUT THE Y2 (RIGHT WING)
C      AXIS.
C*****
C SUBROUTINE ACTRM2
COMMON /ABD4/ PSINEX,THENEW,PHINEX,PSICLD,THEOLD,PHIOLD
COMMON /AEE5/ FWDACC,AN,MCONT
COMMON /ABF1/ U1,V1,W1,U2,V2,W2,U3,V3,W3
COMMON /AEF2/ UD1,VD1,WD1,UD2,VD2,WD2,UD3,VD3,WD3
ANG=THENEW
U3=COS(ANG)*U2-SIN(ANG)*W2
V3=V2
W3=SIN(ANG)*U2+CCS(ANG)*W2
UD3=COS(ANG)*UD2-SIN(ANG)*WD2
VD3=VD2
WD3=SIN(ANG)*UD2+CCS(ANG)*WD2
RETURN
END

```



```

C*****
C SUBROUTINE ACTRM3
C*****
C***** ROTATES AIRCRAFT ABOUT THE X3 (NOSE) AXIS *****
C*****
SUBROUTINE ACTRM3
COMMON /ABD4/ PSINew,THENew,PHINew,PSIOld,THEOld,PHIOld
COMMON /AEE5/ FWCACC,AN,MCONT
COMMON /ABF1/ U1,V1,W1,U2,V2,W2,U3,V3,W3
COMMON /AEF2/ CD1,VD1,WD1,UD2,VD2,WD2,UD3,VD3,WD3
COMMON /AEF3/ UAC,VAC,WAC,UDOTAC,VDOTAC,WDOTAC,P,Q,R
ANG=PHINew
UAC=U3
VAC=cos(ANG)*V3+sin(ANG)*W3
WAC=-sin(ANG)*V3+cos(ANG)*W3
UDOTAC=UD3
VDCTAC=cos(ANG)*VD3+sin(ANG)*WD3
WDCTAC=-sin(ANG)*VD3+cos(ANG)*WD3
RETURN
END

```

```

C*****
C SUBROUTINE ACRVEL
C*****
C***** ACRVEL (AIRCRAFT ROTATIONAL VELOCITIES) COMPUTES THE *****
C***** ROTATIONAL VELOCITIES IMPOSED ON THE *****
C***** AIRCRAFT, IN THE ROTATIONAL OR AIRCRAFT COORDINATE *****
C***** SYSTEM. (X ALONG THE NOSE, Y OUT THE RIGHT WING, *****
C***** Z OUT THE BOTTOM OF THE AIRCRAFT.) *****
C*****
SUBROUTINE ACRVEL
COMMON /ABD3/ PSIDOT,THEDOT,PHIDOT
COMMON /ABD4/ PSINew,THENew,PHINew,PSIOld,THEOld,PHIOld
COMMON /AEE5/ FWCACC,AN,MCONT
COMMON /ABF3/ UAC,VAC,WAC,UDOTAC,VDOTAC,WDOTAC,P,Q,R
ANG1=THENew
ANG2=PHINew
P=PHIDOT-sin(ANG1)*PSIDOT
Q=cos(ANG2)*THEDOT+cos(ANG1)*sin(ANG2)*PSIDOT
R=-sin(ANG2)*THEDOT+cos(ANG1)*cos(ANG2)*PSIDOT
RETURN
END

```



```

C*****
C SUBROUTINE ACACC (AIRCRAFT ACCELERATIONS) PUTS TOGETHER THE LINEAR
C ACCELERATIONS RESULTING FROM TRANSLATION THROUGH
C SPACE AND THE ROTATIONAL ACCELERATIONS RESULTING
C FROM MANUVERING. WHAT APPEARS BELOW IS THE LHS
C OF THE CLASSIC EQUATIONS OF MOTION FOR AN AIRCRAFT.
C*****
SUBROUTINE ACACC
COMMON /ABC2/ XNEW,DELX,XTEMP,XCLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /ABE3/ AX,AY,AZ
COMMON /ABE5/ FWDACC,AN,MCONT
COMMON /ABE3/ UAC,VAC,WAC,UDOTAC,VDOTAC,WDOTAC,P,Q,R
AX=UDOTAC-VAC*E+WAC*Q
AY=VDOTAC+UAC*E-WAC*P
AZ=WDOTAC-UAC*Q+VAC*P
A=-VAC*R
E=WAC*Q
C=UAC*R
D=-WAC*P
E=-UAC*Q
F=WAC*P
RETURN
END

```

```

C*****
C SUBROUTINE ACLMIS (AIRCRAFT LIMITATIONS) COMPARES THE
C ACCELERATIONS RESULTING FROM A PROPOSED MOVE ALONG A
C FLIGHT-PATH TRAJECTORY TO THE PHYSICAL LIMITATIONS OF
C THE AIRCRAFT. THIS IS ESSENTIALLY THE RHS OF THE
C EQUATIONS OF MOTION. IF THE TRAJECTORY ACCELERATIONS
C ARE LARGER THAN ALLOWABLE THEY ARE ADJUSTED DOWNWARD
C IN MAGNITUDE UNTIL THE RHS AND LHS OF THE EQUATIONS
C OF MOTION ARE EQUATED.
C*****
SUBROUTINE ACLMIS
COMMON /ABE3/ AX,AY,AZ
COMMON /ABE4/ THMAX,ANMAX,CLMAX,GEE,TO,SHOSSL,WL,CDO,CDK,AREA,DTDH
COMMON /ABE5/ FWDACC,AN,MCONT
CALL THRUST
IF (AN.LT.ANMAX.AND.FWDACC.LT.THMAX) CALL BOOSTR
C IF (AN.GT.ANMAX.OR.FWDACC.GT.THMAX) CALL REDUCR
RETURN
END

```



```

C*****
C SUBROUTINE BOOSTR
C          BCCSTR INCREASES THE LINEAR ACCELERATIONS UNTIL
C          MAXIMUM ALLOWABLE PERFORMANCE IS REACHED
C*****
      SUBROUTINE BOOSTR
      COMMON /ABC2/ XNEW,DELY,XTEMP,XCLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1 ZTEMP,ZOLD
      COMMON /ABC4/ AG1,AH1,AI1,AJ1,AJ2
      COMMON /AEE1/ UDOT,VDOT,WDOT
      COMMON /AEE3/ AX,AY,AZ
      COMMON /AEE4/ THMAX,ANMAX,CLMAX,GEE,TO,RHOSSL,WL,CDO,CDK,AREA,DTDH
      COMMON /AEE5/ FWDACC,AN,MCONT
      MCCNT=1
10      CONTINUE
      I=0
      IF (AG1.EQ.DELEX.OR.AG1.EQ.0.) GO TO 20
      IF (ABS(AG1).LT.AES(AH1).AND.AES(AG1).LT.ABS(AI1)) I=1
      IF (I.NE.1) GO TO 20
      UDOT=1.02*UDOT
20      CONTINUE
      IF (AH1.EQ.DELEX.OR.AH1.EQ.0.) GO TO 30
      IF (ABS(AH1).LT.AES(AG1).AND.AES(AH1).LT.ABS(AI1)) I=2
      IF (I.NE.2) GO TO 30
      VDOT=1.02*VDOT
30      CONTINUE
      IF (AI1.EQ.DELEX.OR.AI1.EQ.0.) GO TO 40
      IF (ABS(AI1).LT.AES(AG1).AND.AES(AI1).LT.ABS(AH1)) I=3
      IF (I.NE.3) GO TO 40
      WDOT=1.02*WDOT
40      CONTINUE
      IF (AG1.EQ.DELEX.OR.AG1.EQ.0.) GO TO 50
      IF (ABS(AG1).EQ.AES(AI1).AND.AES(AG1).LT.ABS(AH1)) I=4
      IF (I.NE.4) GO TO 50
      UDOT=1.02*UDOT
      WDOT=1.02*WDOT
50      CONTINUE
      IF (AH1.EQ.DELEX.OR.AH1.EQ.0.) GO TO 60
      IF (ABS(AG1).EQ.AES(AH1).AND.AES(AG1).LT.ABS(AI1)) I=5
      IF (I.NE.5) GO TO 60
      UDOT=1.02*UDOT
      VDOT=1.02*VDOT
60      CONTINUE
      IF (AI1.EQ.DELEX.OR.AI1.EQ.0.) GO TO 70
      IF (ABS(AH1).EQ.AES(AI1).AND.AES(AH1).LT.ABS(AG1)) I=6
      IF (I.NE.6) GO TO 70
      VDOT=1.02*VDOT
      WDOT=1.02*WDOT
70      CONTINUE
      CALL SMLCAL
      CALL THRUST
      IF (AN.GE.ANMAX.OR.FWDACC.GE.THMAX) GO TO 80
      GO TO 10
80      CONTINUE
      IF (AN.GT.ANMAX) AN=ANMAX
      IF (FWDACC.GT.THMAX) FWDACC=THMAX
      RETURN
      END

```



```

C*****
C SUBROUTINE REDUCR      REDUCR IS CALL FROM ACLM TS AND REDUCES THE "NEW"
C                        VELOCITY AND ULTIMATELY POSITION TO GET LOADS
C                        ON THE AIRCRAFT WITHIN LIMITS.
C*****
C SUBROUTINE REDUCR
COMMON /ABC3/ VELCN, VELOT, VELCC, N, UNEW, UTEMP, UOLD, VNEW, VTEMP, VOLD,
1 WNEW, WTEMP, WGLE
COMMON /ABC4/ AG1, AH1, AI1, AJ1, AJ2
COMMON /AED2/ VSTALL, VMAX, VMAX1, VMAX2, DELTEE, DELTV, DELTZ
COMMON /AEE3/ AX, AY, AZ
COMMON /AEE4/ THMAX, ANMAX, CLMAX, GEE, TO, RHOSL, L, CDO, CDK, AREA, DTDH
COMMON /AEE5/ FWDACC, AN, MCONT
COMMON /LCC/ X1, Y1, Z1, V1, ITR, ITR2
MCONT=1
10 CONTINUE
VELON=VELCN-DELTV
A=VSTALL+2.*DELTV
IF (VELON.GE.A) GO TO 20
ANG=ACOS (AG1/AJ2)
AJ2=1.1*AJ2
AG1=COS (ANG) *AJ2
AH1=SIN (ANG) *AJ2
VELON=V1
20 CONTINUE
N=0
CALL MVELCS
CALL MVDCTS
CALL SMLCAL
CALL THRUST
IF (AN.LE.ANMAX.AND.FWDACC.LE.THMAX) GO TO 30
GO TO 10
30 CONTINUE
RETURN
END

```



```

C*****
C SUBROUTINE SMLCAL (SMALL CALL) IS A SMALLER VERSION OF (BIG CALL) AND
C                      USED IN THE ITERATIVE LCOFS OF (AIRCRAFT LIMITATIONS)
C*****
C SUBROUTINE SMLCAL
C   CALL MVDOTS
C   CALL DELVEC
C   CALL TMEPNT
C   CALL TMPVEL
C   CALL INBTRM
C   CALL ANGLES
C   CALL ROTRAT
C   CALL ACTRM1
C   CALL ACTRM2
C   CALL ACTRM3
C   CALL ACRVEL
C   CALL ACACC
C   RETURN
C   END

```

```

C*****
C SUBROUTINE THRUST THRUST IS USED BY (AIRCRAFT LIMITATIONS) TO COMPUTE
C                      THE AVAILABLE FOR*AND ACCELERATION BASED UPON THRUST
C                      DRAG, G-LOADING, POSITION IN SPACE AND GRAVITY.
C*****
C SUBROUTINE THRUST
C   COMMON /ABC2/ XNEW,DELX,XTEMP,XCLD,YNEW,DELY,YTEMP,YOLD,CNEW,DELZ,
1ZTEMP,ZOLD
C   COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1VNEW,VTEMP,VOLD
C   COMMON /ABC4/ PSINW,THNEW,PHNEW,PSIOLD,THEOLD,PHIOLD
C   COMMON /ABC5/ AX,AY,AZ
C   COMMON /AEE3/ THMAX,AMAX,CLMAX,GEE,TO,RHOSSL,WL,CDO,CDK,AREA,DTDH
C   COMMON /AEE5/ FWDACC,AN,MCONT
C   ANG=THNEW
C   THSQD=AX/GEE
C   AN=SQRT(AY**2+AZ**2)/GEE
C   SIGMA=(1+LTDH*ZTEMP/TO)**4.257
C   RHOALT=RHOSSL*SIGMA
C   CL=(2.*AN*WL)/(RHOALT*VELCT**2)
C   DRAG=AN*((CDO+CDK*CL**2)/CL)
C   FWDACC=SIN(ANG)+THSQD+DRAG
C   RETURN
C   END

```



```

C*****
C SUBROUTINE XCENGE
C      ONCE THE EQUATIONS OF MOTION ARE SATISFIED AT THE
C      PROPOSED LOCATION (TEMP) THE TEMP LOCATION BECOMES
C      THE CURRENT LOCATION (OLD).
C*****
      SUBROUTINE XCHNGE
      COMMON /AEC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1 ZTEMP,ZOLD
      COMMON /AEC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1 WNEW,WTEMP,WOLD
      COMMON /AED4/ PSINEW,THENEW,PHINEW,PSIOLD,THEOLD,PHIOLD
      XOLD=XTEMP
      YOLD=YTEMP
      ZOLD=ZTEMP
      UOLD=UTEMP
      VOLD=VTEMP
      WOLD=WTEMP
      PSIOLD=PSINEW
      THEOLD=THENEW
      PHIOLD=PHINEW
      VELOC=VELOT
      RETURN
      END

```

```

C*****
C SUBROUTINE WAYPNT
C      THE RELEVANT PARAMETERS FOR THE CURRENT POINT ARE
C      ASSIGNED.
C*****
      SUBROUTINE WAYPNT
      COMMON /AEC1/ WX(1000),WY(1000),WZ(1000),WU(1000),WV(1000),WW(1000),
1 WH(1000),WP(1000),WG(1000),I(1000),ICT
      COMMON /AEC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1 ZTEMP,ZOLD
      COMMON /AEC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1 WNEW,WTEMP,WOLD
      COMMON /AED2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTU,DELTAZ
      COMMON /AED4/ PSINEW,THENEW,PHINEW,PSIOLD,THEOLD,PHIOLD
      COMMON /AEE5/ FWDACC,AN,MCONT
      WX(ICT)=XOLD
      WY(ICT)=YOLD
      WZ(ICT)=ZOLD
      WU(ICT)=UOLD
      WV(ICT)=VOLD
      WW(ICT)=WOLD
      WH(ICT)=-PSINEW
      WP(ICT)=THENEW
      WG(ICT)=-PHINEW
      T(ICT)=AN
      T(ICT)=T(ICT-1)+DELTEE
      RETURN
      END

```



```

C*****
C SUBROUTINE FILE      THE CURRENT POINT IS IMMEDIATELY SAVED.
C*****
SUBROUTINE FILE
COMMON /ABC1/  XX(1000),YY(1000),ZZ(1000),WU(1000),WV(1000),WW(1000
1),WH(1000),WP(1000),WR(1000),WG(1000),T(1000),ICT
COMMON /ABC2/  XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /LOC/  X1,Y1,Z1,I1,ITR,LIR2
IF (XOLD.NE.XNEW.OR.YOLD.NE.YNEW.OR.ZOLD.NE.ZNEW) GO TO 10
WRITE (11,20) T(ICT),XX(ICT),YY(ICT),ZZ(ICT),WU(ICT),WV(ICT),WW(IC
1T),WH(ICT),WP(ICT),WR(ICT),WG(ICT),LIR
RETURN
10 CONTINUE
LIR=0
WRITE (11,20) T(ICT),XX(ICT),YY(ICT),ZZ(ICT),WU(ICT),WV(ICT),WW(IC
1T),WH(ICT),WP(ICT),WR(ICT),WG(ICT),LIR
RETURN
20 FORMAT (11(E10.4,1X),I3)
END

```

```

C*****
C SUBROUTINE GRAFIX    THE CURRENT POINT ALONG THE TRAJECTORY TO THE
C                      DESIRED WAYPOINT IS PLOTTED.
C*****
SUBROUTINE GRAFIX
COMMON /ABC1/  XX(1000),YY(1000),ZZ(1000),WU(1000),WV(1000),WW(1000
1),WH(1000),WP(1000),WR(1000),WG(1000),T(1000),ICT
COMMON /MN1/  MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
PX1=XX(ICT-1)
PX2=XX(ICT)
PY1=YY(ICT-1)
PY2=YY(ICT)
CALL WIN (MINX,MAXX,18000.,12000.,0)
CALL GVECT (0.,PX1,PY1)
CALL GVECT (1.,PX2,PY2)
CALL GSFECE
RETURN
END

```



```

C*****
C SUBROUTINE ERRCHK
C          ERRCHK CHECKS THE DESIRED WAYPOINT AGAINST GAME
C          LIMITATIONS. WHENEVER POSSIBLE, CORRECTIONS ARE
C          AUTOMATIC.
C*****
      SUBROUTINE ERRCHK (IERR)
      COMMON /LOC/ X1,Y1,Z1,V1,LTR,LTR2
      COMMON /PAR4/ APFMAX,HTMIN,HTMAX,POPMIN,BAALT,BRMIN,BRMAX,BRRMIN,B
1 BRMAX,RTD,AMTF
      COMMON /TAR/ TARGX,TARGY
      COMMON /ABC1/ WX(1000),WY(1000),WZ(1000),WU(1000),WV(1000),WW(1000
1),WH(1000),WP(1000),WR(1000),WG(1000),I(1000),ICT
      DX=TARGX-X1
      DY=TARGY-Y1
      DIST=SQRT(DX**2+DY**2)
      IF (DIST.LE.PCFMIN) GO TO 10
      IF (Z1.GT.APPMAX) Z1=APPMAX
      IF (Z1.LT.HTMIN) Z1=HTMIN
10  CONTINUE
      IF (LTR.NE.65.OR.LTR2.NE.65) GC TO 20
      IF (Z1.LT.BAALT) Z1=BAALT
20  CONTINUE
      IF (LTR.NE.66.OR.LTR2.NE.66) GC TO 60
      IF (Z1.LT.BRMIN) Z1=BRMIN
      IF (Z1.GT.BRMAX) Z1=BRMAX
      IF (DIST.LE.BRRMAX.AND.DIST.GE.BRRMIN) GO TO 30
      IF (DIST.GT.BRRMAX) IERR=9
      IF (DIST.LT.BRRMIN) IERR=13
30  RETURN
      CONTINUE
      TGIHDG=ATAN2(DY,JY)*RTD
      IF (TGIHDG.LT.0.) TGIHDG=TGIHDG+360.
      DX=X1-WX(ICT)
      DY=Y1-WY(ICT)
      ACHDG=ATAN2(DY,DX)*RTD
      IF (ACHDG.LT.0.) ACHDG=ACHDG+360.
      HDGLMT=AES(TGIHDG-ACHDG)
      IF (HDGLMT.LE.5.) GO TO 40
      IERR=10
      RETURN
40  CONTINUE
      DIST=SQRT(DX**2+DY**2)
      SPEED=SQRT(WU(ICT)**2+WV(ICT)**2+WW(ICT)**2)
      DT=DIST/SPEED
      IF (DT.GE.2.33) GC TO 50
      IERR=11
      RETURN
50  CONTINUE
60  CONTINUE
      RETURN
      END

```



```

C*****
C SUBROUTINE AIMPT      THIS SUBROUTINE MARKS A LINE FROM THE CURRENT
C                        AIRCRAFT POSITION TO THE TARGET AS A USER AID IN
C                        ALIGNING BOMB RELEASE POINT.
C*****
      SUBROUTINE AIMPT
      COMMON /AEC1/ WX(1000),WY(1000),WZ(1000),WU(1000),WV(1000),WW(1000)
      1),WH(1000),WP(1000),WR(1000),WG(1000),T(1000),ICT
      COMMON /TAR/ TARGX,TARGY
      REAL*4 TX,TY,TX1,TY1
      CALL MIN (MINX,MAXX,18000.,12000.,0)
      TX=WX(ICT)
      TY=WY(ICT)
      TX1=TARGX
      TY1=TARGY
      CALL GSLT (1)

C-----
C MCVE AND DRAW DASHED LINE TO CURRENT WAYPT COORDINATES
C-----
      CALL GBVECT (0.,TX,TY)
      CALL GBVECT (1.,TX1,TY1)
      CALL GSLT (0)
      CALL GSPECE
      RETURN
      END

```



```

C*****
C SUBROUTINE ELFIN
C      THIS SUBROUTINE REQUESTS THE USER OPTIONS AND CLOSES
C      THE GRAPHIC SUBROUTINES.
C*****
      SUBROUTINE ELFIN (LAST)
      COMMON /CPT/ IGUN,IPNCH,TEXT,ISAM,IMP,KER,IMPZ
      COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
C-----
C CLOSE GRAPHICS ROUTINES
C-----
      CALL DSTERM
      IF (LAST.EQ.0) GO TO 20
10     CONTINUE
      WRITE (6,40)
      READ (5,*) FIN
      CALL PRTCMS ('CLASCSN ')
      IF (FIN.GT.2.OR.FIN.LT.1) GO TO 10
20     CONTINUE
      WRITE (6,50)
      WRITE (6,60)
      READ (5,*) IPNCH
      CALL PRTCMS ('CLASCSN ')
      IF (IPNCH.EQ.0) RETURN
      IF (IPNCH.EQ.2) GO TO 30
      WRITE (6,70)
      READ (5,*) TEXT
      CALL PRTCMS ('CLASCSN ')
      IF (IPNCH.EQ.1) RETURN
30     CONTINUE
      WRITE (6,90)
      READ (5,*) ISAM
      CALL PRTCMS ('CLASCSN ')
      IF (ISAM.LT.1.OR.ISAM.GT.7) GO TO 30
      RETURN
40     FORMAT (51H ARE YOU FINISHED WITH THIS FLIGHTPATH: 0=NO; 1=YES)
50     FORMAT (43H OUTPUT FILE: 0=NO OUTPUT; 1=POOL FILE ONLY)
60     FORMAT (51H                2=MICE FILE ONLY; 3=POOL & MICE FILES)
70     FORMAT (49H EXTENDED OUTPUT: 0=NOT WANTED; 1=EXTENDED OUTPUT)
80     FORMAT (29H MISSILE TYPE BETWEEN 1 AND 7)
      END

```



```

C*****
C SUBROUTINE PRESET
C          THIS SUBROUTINE WRITES THE P001 AND MICE II FILES
C          FOR THE USER.
C*****
C SUBROUTINE PRESET
COMMON /AEC1/ WX(1000), WY(1000), WZ(1000), WU(1000), WV(1000), WW(1000)
1) WH(1000), WP(1000), WR(1000), WG(1000), I(1000), ICT
COMMON /PAR2/ XGUN(7), YGUN(7), ZGUN(7), XSAM, YSAM, ZSAM, GR(7)
COMMON /PAR4/ AEFMAX, HTMIN, HTMAX, BOPMIN, EALDT, BRMIN, BRMAX, BRMIN, B
1BRMAX, BTE, AMTF
COMMON /OPT/ IGUN, IENCH, IEXT, ISAM, IMEXER, IMPZ
COMMON /NAM/ INAM1(2), INAM2(2), IUN3, IANR, IDEST
DIMENSION X10(200), X10(200), VEL(200)
DIMENSION RCSTAB(1000), TINK(10)
DIMENSION VAT1N2(200), VAT3(200), VAT5(200)
C-----
C VULNERABLE AREA TABLE VS TYPE 1 AND 2 WEAPONS
C-----
DATA VAT1N2/2*.4645,6*7.107,2*.6568,6*5.551,2*.6968,6*5.574,2*.656
18*.7.357,2*.6968,6*5.574,2*.6568,6*5.551,2*.6968,6*5.574,2*.6568,
26*.7.357,2*.6968,6*5.574,2*.4645,6*.7432,2*.6568,6*2.358,2*.4645,6*
33*.298,2*.6568,6*2.858,2*.4645,6*.7432,2*.6568,6*2.358,2*.4645,6*3.
4298,2*.6568,6*2.858,2*.6568,6*5.551,2*.6968,6*5.574,2*.6568,6*7.35
57,2*.6968,6*5.574,2*.6568,6*5.551,2*.6968,6*5.574,2*.6568,6*7.35
62*.6968,6*5.574,2*.4645,6*7.107/
C-----
C VULNERABLE AREA TABLE VS TYPE 3 WEAPONS
C-----
DATA VAT3/2*12.54,6*13.47,2*9.853,6*10.51,2*9.853,6*11.15,2*12.64,
16*14.78,2*9.853,6*11.15,2*9.853,6*10.51,2*9.853,6*11.15,2*12.64,6*
214.78,2*9.853,6*11.15,6*1.394,2*4.762,6*.240,2*5.342,6*7.432,2*4.
3762,6*.240,6*1.394,2*4.762,6*.240,2*5.342,6*7.432,2*4.762,6*.24
40,2*9.853,6*10.51,2*9.853,6*11.15,2*12.64,6*14.78,2*9.853,6*11.15,
52*9.853,6*10.51,2*9.853,6*11.15,2*12.64,6*14.78,2*9.853,6*11.15,2*
612.54,6*13.47/
C-----
C VULNERABLE AREA TABLE VS TYPE 5 WEAPONS
C-----
DATA VAT5/8*55.37,8*43.22,8*47.10,8*62.53,8*47.10,8*43.22,8*47.10,
18*62.53,8*47.10,8*5.761,8*27.45,8*33.07,8*27.45,8*5.761,8*27.45,8*
233.07,8*27.45,8*5.761,8*47.10,8*62.53,8*47.10,8*43.22,8*47.10,8*62
3.53,8*47.10,8*55.37/
C-----
C RADAR CROSS SECTION TABLE
C-----
DATA RCSTAB/19*1000.,19*100.,19*10.,19*10.,19*10.,19*100.,19*1000.
1,867*0.0/
C-----
C CALL ERRSET SUPPRESSES ANY POSSIBLE UNDERFLOW PROBLEMS THAT MAY
C RESULT FROM MANIPULATION OF SCENARIO PARAMETERS.
C-----
CALL ERRSET (208,50,-1,1,1)
JAM=0
C-----
C CARD 2 TIME INCREMENT CALCULATION
C-----
TINC=T(ICT)/1000.+6.0008
C-----
C CARD 6 TIME INCREMENT CALCULATIONS
C-----
TINKI=0
DO 10 I=1,9
TINK(I)=TINKI+T(ICT)/10
TINKI=TINK(I)
10 CONTINUE
C-----
C FOLLOWING GENERATES VELOCITY MAGNITUDE AT EACH POINT
C-----
DO 20 I=1,ICT
VEL(I)=SQRT(WU(I)**2+WV(I)**2+WW(I)**2)
20 CONTINUE
C-----
C/////*****PUNCH PROGRAM*****/////
C-----

```



```

C OPTION TO PUNCH THE P001 CARD DECK OR MICE-II CARD DECK
C-----
C IF (IPNCH.EQ.2) GC TC 40
C-----
C COMMENCE PUNCHED OUTPUT OF THE P001 CARD DECK.
C-----
C THE JCL CARDS.
C-----
WRITE (18,70) INAM1(1),INAM1(2),IUNR,IANE,INAM1(1),INAM1(2),IUNR
IF (IDEST.EQ.2) WRITE (18,80) IUNR
WRITE (18,90)
WRITE (18,100)
WRITE (18,110)
WRITE (18,120)
WRITE (18,130)
WRITE (18,140)
WRITE (18,150)
WRITE (18,160)
WRITE (18,170)
C-----
C LEADING BLANK DATA CARD SIGNIFIES RADAR MASKING ANGLE OF ZERO.
C-----
C THE OUTPUT TITLE CARD.
C-----
WRITE (18,200)
C-----
C CARD 2
C-----
WRITE (18,180)
WRITE (18,210) I(1CT),TINC
C-----
C THE 2A CARDS (MILESTONES).
C-----
DO 30 I=1,1CT
WU(I)=WU(I)*AMTF
WV(I)=WV(I)*AMTF
WW(I)=WW(I)*AMTF
WRITE (18,220) T(I),WX(I),WY(I),WZ(I),WU(I),WV(I),WW(I),WH(I),WF(I)
1) ,WR(I)
30 CONTINUE
C-----
C CARD 3 (GUN EMPLACEMENT CARD).
C-----
WRITE (18,240)
WRITE (18,250) XGUN(1),YGUN(1),ZGUN(1)
C-----
C CARD 4 (GUN TYPE).
C-----
WRITE (18,230)
WRITE (18,260)
C-----
C CARD 5
C-----
WRITE (18,190)
WRITE (18,270)
C-----
C CARD 6
C-----
WRITE (18,280)
WRITE (18,290) (TINK(I),I=1,9)
C-----
C CARD 7 (VULNERABLE AREA TABLE VS TYPE 1 AND 2 WEAPONS)
C-----
WRITE (18,300)
WRITE (18,310)
WRITE (18,320) (VAT1N2(I),I=1,208)
C-----
C CARD 12 (EXECUTE RUN).
C EXTENDED OUTPUT OPTION
C-----
WRITE (18,330)
IF (IEXT.EQ.1) WRITE (18,350)
IF (IEXT.EQ.1) WRITE (18,340)
C-----

```


C THE REMAINDER OF THE CARDS INTRODUCE NEW GUN LOCATIONS, GUN TYPES
C AND VULNERABLE AREA TABLES TO BE EXECUTED BY THE PROGRAM.

WRITE (18,240)
WRITE (18,250) XGUN(2),YGUN(2),ZGUN(2)

C EXTENDED OUTPUT OPTION

WRITE (18,330)
IF (IEXT.NE.1) WRITE (18,350)
IF (IEXT.EQ.1) WRITE (18,340)
WRITE (18,240)
WRITE (18,250) XGUN(3),YGUN(3),ZGUN(3)
WRITE (18,230)
WRITE (18,360)

C EXTENDED OUTPUT OPTION

WRITE (18,330)
IF (IEXT.NE.1) WRITE (18,350)
IF (IEXT.EQ.1) WRITE (18,340)
WRITE (18,240)
WRITE (18,250) XGUN(4),YGUN(4),ZGUN(4)

C EXTENDED OUTPUT OPTION

WRITE (18,330)
IF (IEXT.NE.1) WRITE (18,350)
IF (IEXT.EQ.1) WRITE (18,340)
WRITE (18,240)
WRITE (18,250) XGUN(5),YGUN(5),ZGUN(5)
WRITE (18,230)
WRITE (18,370)

C CARD 7 (VULNERABLE AREA TABLE VS TYPE 3 WEAPONS)

WRITE (18,300)
WRITE (18,410)
WRITE (18,320) (VAT3(I),I=1,208)

C EXTENDED OUTPUT OPTION

WRITE (18,330)
IF (IEXT.NE.1) WRITE (18,350)
IF (IEXT.EQ.1) WRITE (18,340)
WRITE (18,240)
WRITE (18,250) XGUN(6),YGUN(6),ZGUN(6)
WRITE (18,230)
WRITE (18,380)

C EXTENDED OUTPUT OPTION

WRITE (18,330)
IF (IEXT.NE.1) WRITE (18,350)
IF (IEXT.EQ.1) WRITE (18,340)
WRITE (18,240)
WRITE (18,250) XGUN(7),YGUN(7),ZGUN(7)
WRITE (18,230)
WRITE (18,390)

C CARD 7 (VULNERABLE AREA TABLE VS TYPE 5 WEAPON)

WRITE (18,300)
WRITE (18,420)
WRITE (18,320) (VAT5(I),I=1,208)

C EXTENDED OUTPUT OPTION

WRITE (18,330)
IF (IEXT.NE.1) WRITE (18,350)
IF (IEXT.EQ.1) WRITE (18,340)
WRITE (18,400)
IF (IPNCH.EQ.1) RETURN

C FCNMT STATEMENTS


```

40      CONTINUE
C-----
C COMMENCE PUNCHED OUTPUT OF THE MICE-II CARD DECK
C-----
C CHECK FOR VALID MISSILE DESIGNATION NUMBER FOR MICE II
C-----
      IF (ISAM.LE.7.AND.ISAM.GE.1) GO TO 50
      RETURN
C-----
C THE JCL CARDS
C-----
50      WRITE (17,430) INAM2(1),INAM2(2),IUNR,IANS,INAM2(1),INAM2(2),IUNR
      IF (IDEST.EQ.2) WRITE (17,440) IUNR
      WRITE (17,450)
      WRITE (17,460)
      WRITE (17,470)
      WRITE (17,480)
C-----
C THE OUTPUT TITLE CARDS
C-----
      WRITE (17,490)
      WRITE (17,500)
C-----
C THE PROBLEM RUN INPUT CARDS, CHECKING FOR TYPE OF MISSILE
C TC INPUT TO MICE II
C-----
      WRITE (17,510)
      IF (ISAM.EQ.1) WRITE (17,580)
      IF (ISAM.EQ.2) WRITE (17,590)
      IF (ISAM.EQ.3) WRITE (17,700)
      IF (ISAM.EQ.4) WRITE (17,710)
      IF (ISAM.EQ.5) WRITE (17,720)
      IF (ISAM.EQ.6) WRITE (17,730)
      IF (ISAM.EQ.7) WRITE (17,740)
C-----
C THE PSSK CALCULATION CARDS
C-----
      WRITE (17,520)
      WRITE (17,530)
C-----
C THE RADAR CROSS SECTION TABLE
C-----
      WRITE (17,540)
      WRITE (17,550)
      WRITE (17,560)
      WRITE (17,570)
      WRITE (17,580) (RCSTAB(I),I=1,133)
C-----
C THE SIMULATION TIME PARAMETERS
C-----
      WRITE (17,590)
      WRITE (17,600)
C-----
C THE MISSILE LAUNCHER LOCATION
C-----
      WRITE (17,610)
      XFSAM=XSAM*AMTF
      YFSAM=YSAM*AMTF
      ZFSAM=ZSAM*AMTF
      WRITE (17,620) XFSAM,YFSAM,ZFSAM
C-----
C THE TARGET TRAJECTORY
C-----
      WRITE (17,630)
      WRITE (17,640) ICT
C-----
C THE MILESTONE CARDS
C-----
      DO 60 I=1,ICT
      WX(I)=WX(I)*AMTF
      WY(I)=WY(I)*AMTF
      WZ(I)=WZ(I)*AMTF
      VEL(I)=VEL(I)*AMTF
      WRITE (17,650) I(I),WX(I),WY(I),WZ(I),VEL(I),WP(I),WR(I),WH(I)
60      CCNTINUE

```



```

C -----
C END OF PUNCHED CARDS
C -----
      WRITE (17,660)
      WRITE (17,670)
C -----
      RETURN
C -----
C /////***** FORMAT STATEMENTS *****/////
C
70  FORMAT (2H//,2A4,6H JOB (,I+,1H,I4,2H),,1H',2A4,1H ,I4,2HP')
80  FORMAT (19H//*MAIN ORG=NP GVM1.,I4,1HP)
90  FORMAT (16H// EXEC PGM=P1AD)
100 FORMAT (42H//STEPLIB DD DISP=SHR,DSN=MSS.F0559.PIPSAV)
110 FORMAT (47H//GC.FTC07F001 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),)
120 FORMAT (41H// ICE=(RECFM=1BS,LRECL=404,BLKSIZE=3236))
130 FORMAT (22H//GO.FTC09F001 DD DUMMY)
140 FORMAT (25H//GC.FTC06F001 DD SYSOUT=A)
150 FORMAT (18H//GO.FTC5F001 DD *)
160 FORMAT (11H1,1,0,0,0,0)
170 FORMAT (2H01)
180 FORMAT (2H02)
190 FORMAT (2H05)
200 FORMAT (42H AIRCRAFT COMBAT SURVIVABILITY SCENARIO)
210 FORMAT (7H0,12,0,,F7.2,9H,1000CO.,,F7.4,1H/)
220 FORMAT (10(F7.1,13))
230 FORMAT (2H04)
240 FORMAT (2H03)
250 FORMAT (3(1X,F7.0),1CH 0.0,360.0)
260 FORMAT (19H0,1,1,1,1,1,0.0,50.0)
270 FORMAT (7H1,1,1,0)
280 FORMAT (2H06)
290 FORMAT (3H1,3,9(1X,F7.3))
300 FORMAT (2H07)
310 FORMAT (49H VULNERABLE AREA TABLE VS TYPE 1 AND 2 WEAPONS)
320 FORMAT (6F8.3)
330 FORMAT (2H12)
340 FORMAT (13H1,1,1,1,1,1,1)
350 FORMAT (13H0,0,0,0,1,1,1)
360 FORMAT (20H0,2,1,1,1,1,0.0,50.0)
370 FORMAT (20H0,3,4,1,1,1,0.0,50.0)
380 FORMAT (20H0,3,3,4,1,1,0.0,50.0)
390 FORMAT (20H0,5,3,2,1,1,0.0,50.0)
400 FORMAT (2H/*)
410 FORMAT (43H VULNERABLE AREA TABLE VS TYPE 3 WEAPONS)
420 FORMAT (42H VULNERABLE AREA TABLE VS TYPE 5 WEAPON)
430 FORMAT (2H//,2A4,6H JOB (,I4,1H,I4,2H),,1H',2A4,1H ,I4,2HP')
440 FORMAT (19H//*MAIN ORG=NP GVM1.,I4,1HP)
450 FORMAT (28H// EXEC PGM=MICE,REGION=180K)
460 FORMAT (44H//STEPLIB DD DSN=MSS.F0559.MICESAV,DISP=SHR )
470 FORMAT (23H//FTC6F001 DD SYSCUT=A)
480 FORMAT (15H//FTC5F001 DD *)
490 FORMAT (2H01)
500 FORMAT (53HA2 3251 *****SURVIVABILITY SCENARIO***** SAM ENCOUNTER)
510 FORMAT (2H02)
520 FORMAT (2H06)
530 FORMAT (9X,1H0,8X,2H0.,9X,1H0,4X,1H0,4X,1H0,3X,2H0.,8X,2H0.)
540 FORMAT (2H08)
550 FORMAT (7X,3H23.,3X,2H19,4X,1H7,4X,1H1)
560 FORMAT (8X,2H0.,7X,3H10.,7X,3H20.,7X,3H30.,7X,3H40.,7X,3H50.,7X,3H
160.//,7X,3H70.,7X,3H80.,7X,3H90.,6X,4H100.,6X,4H110.,6X,4H120.,6X,
24H130.,,6X,4H140.,6X,4H150.,6X,3H160,6X,4H170.,6X,4H180.)
570 FORMAT (6X,4H-90.,6X,4H-60.,6X,4H-30.,8X,2H0.,7X,3H30.,7X,3H60.,7X
1,3H90.)
580 FORMAT (7F10.2,/,7F10.2,/,5F10.2)
590 FORMAT (2H09)
600 FORMAT (7X,3H0.5,7X,3H0.1,7X,3H8.0,7X,3H0.0,8X,2H0.,7X,3H30.,8X,2H
10.)
610 FORMAT (2H11)
620 FORMAT (2X,F9.2,2F10.2)
630 FORMAT (2H12)
640 FORMAT (9X,1H1,2X,I3,4X,1H1,4X,1H0,8X,2H0.,8X,2H0.,8X,2H0.)
650 FORMAT (F7.1,4F10.1,3F6.1,4X,2H0.,4X,2H0.)
660 FORMAT (2H20)
670 FORMAT (2H/*)
680 FORMAT (9X,1H1,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)

```



```

690  FORMAT (9X,1H2,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
700  FORMAT (9X,1H3,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
710  FORMAT (9X,1H4,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
720  FORMAT (9X,1H5,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
730  FORMAT (9X,1H6,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
740  FORMAT (9X,1H7,4X,1H1,4X,1H4,4X,1H1,3X,2H10,4X,1H1,4X,1H0)
      END

```



```

C MAXIMUM THRUST OF THE AIRCRAFT (DIMENSIONLESS)
C-----
C DATA THMAX/0.40/
C-----
C MAXIMUM AIRCRAFT GEE-LOADING (DIMENSIONLESS)
C-----
C DATA ANMAX/6./
C-----
C RHOSL IS THE DENSITY AT STANDARD SEA-LEVEL CONDITIONS. IT HAS
C DIMENSIONS OF (N.S2/M4)
C-----
C DATA RHOSL/1.225/
C-----
C STANDARD SEA-LEVEL TEMPERATURE (DEGREES K)
C-----
C DATA TO/288.16/
C-----
C TEMPERATURE LAPSE RATE FOR THE TROPOSPHERE (DEG K/M)
C-----
C DATA DTDR/-0.5E-3/
C-----
C MAXIMUM WING LOADING (N/M2)
C-----
C DATA WL/4788./
C-----
C SPEED BRAKE AREA: DEFINED AS A PERCENTAGE OF TOTAL WING AREA
C-----
C DATA AREA/.05/
C-----
C MAXIMUM COEFFICIENT OF LIFT
C-----
C DATA CLMAX/1./
C-----
C COEFFICIENT OF DRAG AT ZERO ANGLE OF ATTACK
C-----
C DATA CD0/.015/
C-----
C PROFILE DRAG COEFFICIENT
C-----
C DATA CDK/0.1/
C-----
C MINIMUM ALTITUDE FOR THE GAME (M)
C-----
C DATA HTMIN/60./
C-----
C MAXIMUM ALTITUDE APPROACHING THE TARGET (M)
C-----
C DATA APPMAX/457./
C-----
C MAXIMUM ALTITUDE AFTER ENTERING POP-UP RANGE OF 6 KM. (M)
C-----
C DATA HTMAX/2050./
C-----
C MAXIMUM RANGE (METERS) A POP-UP MAY BE COMMENCED
C-----
C DATA PCPMIN/6000./
C-----
C MINIMUM ALLOWABLE POP-UP ALTITUDE (M)
C-----
C DATA BAALI/1000./
C-----
C BOMB RELEASE MAXIMUM ALTITUDE (M)
C-----
C DATA BRMAX/2000./
C-----
C BOMB RELEASE MINIMUM ALTITUDE (M)
C-----
C DATA BRMIN/100./
C-----
C MAXIMUM BOMB RELEASE RANGE (M)
C-----
C DATA BRRMAX/1000./
C-----
C MINIMUM BOMB RELEASE RANGE (M)
C-----
C DATA BRRMIN/100./

```



```

C-----
C MAXIMUM VELOCITIES BEFORE AND AFTER BOMB RELEASE (M/S)
C-----
C DATA VMAX1/260./,VMAX2/310./
C-----
C STALL VELOCITY (M/S)
C-----
C DATA VSTALL/90./
C-----
C INCREMENT OF VELOCITY CHANGE (5 KNCT INC CONVERTED TO M/S)
C-----
C DATA DELTV/3./
C-----
C THE FOLLOWING ARE TEMPORARY PARAMETERS FOR THE INITIAL CONDITIONS OF
C THE AIRCRAFT. LATER THIS WILL BE ENTERED INTERACTIVELY.
C-----
C DATA WX/C./,WY/6000./,WZ/200./,WU/200./,WV/0./,WW/0./
C DATA WH/1.5708/,WF/0./,WB/0./,WG/0./,T/0./
C-----
C INCREMENT OF TIME CHANGE (SECONDS)
C-----
C DATA DELTEE/1.00/
C-----
C INCREMENT OF ALTITUDE CHANGE (50 FT INC CONVERTED TO METERS)
C-----
C DATA DELTAZ/15.24/
C-----
C TEMPORARY PROGRAM INITIALIZATION
C-----
C DATA XOLD/0./,YOLD/6000./,ZOLD/200./
C DATA UOLD/200./,VOLD/0./,WOLD/0./
C DATA PSICLD/0./,THEOLD/0./,PHICLD/0./
C DATA VELCC/200./
C-----
C ENC

```


APPENDIX H

FLIGHTPATH GENERATOR WITHOUT THE GRAPHICS

```

C *****
C SUBROUTINE BIGCAL 16MAR84
C      T11 IS SET UP AS A TROUBLE-SHOOTING CONVENIENCE.
C      IT IS THE FLIGHTPATH GENERATOR PULLED FROM IBMPI2.
C      WHEN IN USE, EACH SUBROUTINE HAD A WRITE-TO-FILE
C      STATEMENT IN IT FOR OBSERVATION PURPOSES.
C      CHANGES AND COMMENTS ARE CURRENT THROUGH 16MAR84.
C      -- D.A.W. --
C      THIS SUBROUTINE OPERATES AT A LEVEL BELOW MAIN
C      PROGRAM AND ACTS AS A DRIVER FOR THE SUBROUTINES
C      LISTED WITHIN.
C *****
C      SUBROUTINE BIGCAL
C      COMMON /ABC1/  X(1000), Y(1000), Z(1000), XU(1000), YV(1000), ZW(1000)
C      1) AH(1000), AP(1000), AR(1000), AG(1000), I(1000), ICT
C      COMMON /ABC2/  XNEW, DELX, XTEMP, XOLD, YNEW, DELY, YTEMP, YOLD, ZNEW, DELZ,
C      1) ZTEMP, ZOLD
C      COMMON /ABC3/  VELCN, VELOT, VELOC, N, UNEW, UTEMP, UOLD, VNEW, VTEMP, VOLD,
C      1) WNEW, WTEMP, WOLD
C      COMMON /ABC4/  AG1, AH1, AI1, AJ1, AJ2
C      COMMON /ABC5/  VSTALL, VMAX1, VMAX2, DELTES, DELTV, DELTZ
C      COMMON /ABC6/  FWLACC, AN, MCONT
C      N=0
C      ICT=2
C      MCONT=0
C      CALL TSNFRM
C      CALL NEWOLD
C      CALL MVELCS
C      CALL MVDCTIS
C      CALL DELVEC
C      CALL IMPENT
C      CALL IMPVEL
C      CALL INBTM
C      CALL ANGLES
C      CALL ROTPAT
C      CALL ACTEM1
C      CALL ACTEM2
C      CALL ACTEM3
C      CALL AC3VEL
C      CALL ACACC
C      CALL ACLMTS
C      CALL XCHNGE
C      CALL WAYFNT
C      CALL FILE
C      ICT=ICT+1
C      IF (ABS(UTEMP) .GT. ABS(VTEMP) .AND. DELX.EQ.AG1) GO TO 20
C      IF (ABS(VTEMP) .GT. ABS(UTEMP) .AND. DELY.EQ.AH1) GO TO 20
C      GO TO 10
C      CONTINUE
C      RETURN
C      STOP
C      END

```



```

C*****
C SUBROUTINE TRNFRM TRANSFORM TAKES THE REAL VALUE VARIABLES SUPPLIED
C BY THE GRAPHICS SUBROUTINES AND LOADS THEM INTO
C VARIABLES SPECIFIC TO THIS AIRCRAFT TRAJECTORY
C GENERATING PACKAGE.
C*****
SUBROUTINE TRNFRM
COMMON /LCC/ X1,Y1,Z1,V1,ITR,ITR2
COMMON /AEC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /AEC3/ VELCN,VELOT,VELOC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1VNEW,VTEMP,VOLD
COMMON /AEC5/ FWDACC,AN,MCONT
XNEW=X1
YNEW=Y1
ZNEW=Z1
VELON=V1
RETURN
END

```

```

C*****
C SUBROUTINE NEWOLD NEWOLD GENERATES LENGTH VECTOR COMPONENTS BETWEEN
C THE CURRENT WAYPOINT (OLD) AND THE DESIRED WAYPOINT
C USER HAS JUST ENTERED (NEW).
C*****
SUBROUTINE NEWOLD
COMMON /AEC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /AEC4/ AG1,AH1,AI1,AJ1,AJ2
COMMON /AEC5/ FWDACC,AN,MCONT
AG1=XNEW-XOLD
AH1=YNEW-YOLD
AI1=ZNEW-ZOLD
AJ1=SQRT(AG1**2+AH1**2+AI1**2)
AJ2=SQRT(AG1**2+AH1**2)
RETURN
END

```



```

C *****
C SUBROUTINE MVELO3
C MVELO3 (MAP VELOCITY COMPONENTS) DETERMINES THE
C VELOCITY COMPONENTS BETWEEN CURRENT WAYPOINT (OLD)
C AND THE JUST ENTERED WAYPOINT (NEW). THE SUBROUTINE
C CONTAINS A FLAG (N) TO AVOID RESETTING THE VELOCITY
C COMPONENTS UNTIL EITHER A NEW WAYPOINT IS RECEIVED
C OR AN AIRCRAFT LIMITATION WHICH IS FATAL IS REACHED
C IN THE SUBSEQUENT ITERATIVE SUBROUTINES. THE
C COORDINATE SYSTEM IS TERMED MAP, I.E., Y POINTS NORTH
C *****
      SUBROUTINE MVELO3
      COMMON /ABC4/ AG1,AH1,AI1,AJ1,AJ2
      COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1  WNEW,VTEMP,VOLD
      COMMON /ABC5/ FWDACC,AN,MCONT
      M=N
      IF (N.GT.0) GO TO 10
      UNEW=VELCN*(AG1/AJ1)
      VNEW=VELCN*(AH1/AJ1)
      WNEW=VELCN*(AI1/AJ1)
10  CONTINUE
      N=1
      RETURN
      END

```

```

C *****
C SUBROUTINE MVDOTS
C MVDOTS (MAP VELOCITY DOT COMPONENTS) COMPUTES THE
C LINEAR ACCELERATION COMPONENTS
C IN MAP COORDINATES. (Y POINTS NORTH)
C *****
      SUBROUTINE MVDOTS
      COMMON /ABC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1  ZTEMP,ZOLD
      COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1  WNEW,VTEMP,VOLD
      COMMON /ABC4/ AG1,AH1,AI1,AJ1,AJ2
      COMMON /ABC1/ UDOT,VDOT,WDOT
      COMMON /ABC5/ FWDACC,AN,MCONT
      ACC=(VELCN**2-VELCC**2)/(2*AJ1)
      UDOT=ACC*(AG1/AJ1)
      VDOT=ACC*(AH1/AJ1)
      WDOT=ACC*(AI1/AJ1)
      RETURN
      END

```



```

C*****
C SUBROUTINE DELVEC
C DELVEC (DELTA VECTORS) UTILIZES STANDARD DYNAMIC
C EQUATIONS TO GENERATE LINEAR TRANSLATION INCREMENTS.
C IT FURTHER LOOKS TO SEE WHETHER THESE INCREMENTS
C EXCEED THE DISTANCE LEFT TO GO TO THE NEW WAYPOINT
C*****
C SUBROUTINE DELVEC
COMMON /AEC2/ XNEW, DELX, XTEMP, XOLD, YNEW, DELY, YTEMP, YOLD, ZNEW, DELZ,
1ZTEMP, ZOLD
COMMON /AEC3/ VELCN, VELOT, VELOC, N, UNEW, UTEMP, UOLD, VNEW, VTEMP, VOLD,
1VNEW, VTEMP, VOLD
COMMON /AEC4/ AG1, AH1, AI1, AJ1, AJ2
COMMON /AED2/ VSTALL, VMAX, VMAX1, VMAX2, DELTEE, DELTV, DELTZ
COMMON /AEE1/ UECT, VLOT, VDOT
COMMON /AEE5/ FWDACC, AN, MCONT
DELX=XOLD*DELTEE+.5*VDOT*DELTEE**2
DELY=YOLD*DELTEE+.5*VDOT*DELTEE**2
DELZ=ZOLD*DELTEE+.5*VDOT*DELTEE**2
IF (ABS(DELX).GT.ABS(AG1)) DELX=AG1
IF (ABS(DELY).GT.ABS(AH1)) DELY=AH1
IF (ABS(DELZ).GT.ABS(AI1)) DELZ=AI1
RETURN
END

C*****
C SUBROUTINE TMPENT
C TMPENT (TEMPORARY POINT) GENERATES A TEMPORARY
C POSITION IN SPACE BASED UPON THE DELTA VECTORS
C*****
C SUBROUTINE TMPENT
COMMON /AEC2/ XNEW, DELX, XTEMP, XOLD, YNEW, DELY, YTEMP, YOLD, ZNEW, DELZ,
1ZTEMP, ZOLD
COMMON /AEC3/ VELCN, VELOT, VELOC, N, UNEW, UTEMP, UOLD, VNEW, VTEMP, VOLD,
1VNEW, VTEMP, VOLD
COMMON /AEC4/ AG1, AH1, AI1, AJ1, AJ2
COMMON /AED2/ VSTALL, VMAX, VMAX1, VMAX2, DELTEE, DELTV, DELTZ
COMMON /AEE5/ FWDACC, AN, MCONT
XTEMP=XOLD+DELX
YTEMP=YOLD+DELY
ZTEMP=ZOLD+DELZ
IF (DELX.EQ.AG1) XTEMP=XNEW
IF (DELY.EQ.AH1) YTEMP=YNEW
IF (DELZ.EQ.AI1) ZTEMP=ZNEW
RETURN
END

```



```

C*****
C SUBROUTINE TMPEVEL
C TMPEVEL (TEMPORARY VELOCITY COMPONENTS) GENERATES
C THE VELOCITY COMPONENTS ASSOCIATED WITH THE TEMPORARY
C POINT IN SPACE (MAP COORDINATES).
C*****

```

```

SUBROUTINE TMPEVEL
COMMON /ABC2/ XNEW,DELY,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1WNEW,WTEMP,WOLD
COMMON /ABC4/ AG1,AH1,AI1,AJ1,AJ2
COMMON /ABD2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTIV,DELTAV
COMMON /ABE1/ UDOT,VDOT,WDOT
COMMON /ABE5/ FWDACC,AN,MCONT
UTEMP=UOLD+UDOT*DELTEE
VTEMP=VOLD+VDOT*DELTEE
WTEMP=WOLD+WDOT*DELTEE
IF (ABS(UTEMP).GT.ABS(UNEW).AND.MCONT.EQ.1) UTEMP=UNEW
IF (ABS(VTEMP).GT.ABS(VNEW).AND.MCONT.EQ.1) VTEMP=VNEW
IF (ABS(WTEMP).GT.ABS(WNEW).AND.MCONT.EQ.1) WTEMP=WNEW
IF (DELY.EQ.AG1) UTEMP=UNEW
IF (DELY.EQ.AH1) VTEMP=VNEW
IF (DELZ.EQ.AI1) WTEMP=WNEW
IF (DELY.NE.0.) GC TC 10
UDOT=0.
GO TO 20
CONTINUE
10 IF (UTEMP.EQ.UNEW) UDOT=(UTEMP**2-UOLD**2)/(2*DELY)
20 CONTINUE
IF (DELY.NE.0.) GC TC 30
VDOT=0.
GO TO 40
CONTINUE
30 IF (VTEMP.EQ.VNEW) VDOT=(VTEMP**2-VOLD**2)/(2*DELY)
40 CONTINUE
IF (DELZ.NE.0.) GC TC 50
WDOT=0.
GO TO 60
CONTINUE
50 IF (WTEMP.EQ.WNEW) WDOT=(WTEMP**2-WOLD**2)/(2*DELZ)
60 CONTINUE
VELOT=SQRT(UTEMP**2+VTEMP**2+WTEMP**2)
RETURN
END

```

```

C*****
C SUBROUTINE INRTRM
C INRTRM (INITIAL ROTATIONAL TRANSFORMATION) TRANSFORMS
C MAP COORDINATE COMPONENTS TO A SYSTEM USED BY FLIGHT
C DYNAMICISTS (Y POINTS SOUTH AND Z POINTS DOWN)
C*****

```

```

SUBROUTINE INRTRM
COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1WNEW,WTEMP,WOLD
COMMON /ABE1/ UDOT,VDOT,WDOT
COMMON /ABE5/ FWDACC,AN,MCONT
COMMON /ABF1/ U1,V1,W1,U2,V2,W2,U3,V3,W3
COMMON /ABF2/ UC1,VC1,WC1,UC2,VC2,WC2,UC3,VC3,WC3
U1=UTEMP
V1=-VTEMP
W1=-WTEMP
UC1=UDOT
VC1=-VDOT
WC1=-WDOT
RETURN
END

```



```

C*****
C SUBROUTINE ANGLES
C      ANGLES COMPUTES THE ANGLE CHANGE INCREMENTS CAUSED
C      BY MOVEMENT FORWARD TO THE TEMPORARY POINT. PSINew
C      IS THE LEADING CHANGE, REFERENCED TO THE X AXIS.
C      THE NEW IS THE PITCH CHANGE REFERENCED TO THE
C      HORIZONTAL PLANE. PHINew IS THE ROLL INCREMENT ABOUT
C      THE X AXIS.
C*****
SUBROUTINE ANGLES
COMMON /AEC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /AEC3/ VELCN,VELOT,VELCC,N,UNEW,UIEMP,UOLD,VNEW,VTEMP,VOLD,
1WNEW,WTEMP,WOLD
COMMON /AEC4/ AG1,AH1,AI1,AJ1,AJ2
COMMON /AED1/ AA1,AB1,AC1
COMMON /AED2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTV,DELTV
COMMON /ABD4/ PSINew,THENew,PHINew,PSIOLD,THEOLD,PHIOLD
COMMON /AEE5/ FWCACC,AN,MCONT
COMMON /ABF4/ DS,DSE,DSPP
DS=SQRT(DELX**2+DELY**2+DELZ**2)
IF (DELA-NE-0.) GO TO 10
IF (DELX-EQ-0..AND..DELY-LT-0.) PSINew=AA1
IF (DELX-EQ-0..AND..DELY-GT-0.) PSINew=-AA1
GO TO 20
10 CONTINUE
PSINew=-ATAN(DELY/DELX)
IF (DELX-GT-0..AND..DELY-EQ-0.) PSINew=0.
IF (DELX-LT-0..AND..DELY-LT-0.) PSINew=AB1+PSINew
IF (DELX-LT-0..AND..DELY-EQ-0.) PSINew=AB1
IF (DELX-LT-0..AND..DELY-GT-0.) PSINew=PSINew-AB1
20 CONTINUE
THENew=ASIN(DELZ/DS)
TRNRAT=(PSINew-PSIOLD)/DELTEE
IF (PSINew-GT-0.) PHINew=ATAN(ABS(TRNRAT*VELOT/9.7958))
IF (PSINew-LT-0.) PHINew=-ATAN(ABS(TRNRAT*VELOT/9.7958))
IF (PSINew-EQ-PSIOLD) PHINew=PHIOLD
RETURN
END

C*****
C SUBROUTINE ROTRAT
C      ROTRAT (ROTATIONAL RATES) COMPUTES THE ANGLE DOT
C      COMPONENTS IN THE INERTIAL REFERENCE SYS. (THE ONE
C      WITH Y POINTING SOUTH AND Z DOWN).
C*****
SUBROUTINE ROTRAT
COMMON /AEC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /AED2/ VSTALL,VMAX,VMAX1,VMAX2,DELTEE,DELTV,DELTV
COMMON /ABD3/ PSIDOT,THEDOT,PHIDOT
COMMON /AED4/ PSINew,THENew,PHINew,PSIOLD,THEOLD,PHIOLD
COMMON /AEE5/ FWCACC,AN,MCONT
PSIDOT=(PSINew-PSIOLD)/DELTEE
THEDOT=(THENew-THEOLD)/DELTEE
PHIDOT=(PHINew-PHIOLD)/DELTEE
RETURN
END

```



```

C*****
C SUBROUTINE ACTRM1
C      ACTRM1 (AIRCRAFT TRANSFORMATION MATRIX #1) EFFECTS
C      ROTATION OF AIRCRAFT ABOUT THE Z1 AXIS (POS Z
C      POINTS DOWN TO EARTH).
C*****
      SUBROUTINE ACTRM1
      COMMON /AED4/ PSINew,THENew,PHINew,PSIOld,THEOld,PHIOld
      COMMON /AED5/ FWDACC,AN,MCONT
      COMMON /AEP1/ U1,V1,W1,U2,V2,W2,U3,V3,W3
      COMMON /AEP2/ UD1,VD1,WD1,UD2,VE2,WD2,UE3,VD3,WD3
      ANG=PSINew
      U2=cos(ANG)*U1+sin(ANG)*V1
      V2=-sin(ANG)*U1+cos(ANG)*V1
      W2=W1
      UE2=cos(ANG)*UD1+sin(ANG)*VD1
      VD2=-sin(ANG)*UD1+cos(ANG)*VD1
      WD2=WD1
      RETURN
      END

C*****
C SUBROUTINE ACTRM2
C      ACTRM2 ROTATES THE AIRCRAFT ABOUT THE Y2 (RIGHT WING)
C      AXIS.
C*****
      SUBROUTINE ACTRM2
      COMMON /AED4/ PSINew,THENew,PHINew,PSIOld,THEOld,PHIOld
      COMMON /AED5/ FWDACC,AN,MCONT
      COMMON /AEP1/ U1,V1,W1,U2,V2,W2,U3,V3,W3
      COMMON /AEP2/ UD1,VD1,WD1,UD2,VE2,WD2,UE3,VD3,WD3
      ANG=THENew
      U3=cos(ANG)*U2-sin(ANG)*W2
      V3=V2
      W3=sin(ANG)*U2+cos(ANG)*W2
      UE3=cos(ANG)*UE2-sin(ANG)*WD2
      VD3=VD2
      WD3=sin(ANG)*UE2+cos(ANG)*WD2
      RETURN
      END

C*****
C SUBROUTINE ACTRM3
C      ACTRM3 ROTATES AIRCRAFT ABOUT THE X3 (NOSE) AXIS
C*****
      SUBROUTINE ACTRM3
      COMMON /AED4/ PSINew,THENew,PHINew,PSIOld,THEOld,PHIOld
      COMMON /AED5/ FWDACC,AN,MCONT
      COMMON /AEP1/ U1,V1,W1,U2,V2,W2,U3,V3,W3
      COMMON /AEP2/ UD1,VD1,WD1,UD2,VE2,WD2,UE3,VD3,WD3
      COMMON /AEP3/ UAC,VAC,WAC,UDOTAC,VDOTAC,ADOTAC,P,Q,R
      ANG=PHINew
      UAC=U3
      VAC=cos(ANG)*V3+sin(ANG)*W3
      WAC=-sin(ANG)*V3+cos(ANG)*W3
      UDOTAC=UE3
      VDOTAC=cos(ANG)*VD3+sin(ANG)*WD3
      ADOTAC=-sin(ANG)*VD3+cos(ANG)*WD3
      RETURN
      END

```



```

C *****
C SUBROUTINE ACRVEL
C      ACRVEL (AIRCRAFT ROTATIONAL VELOCITIES) COMPUTES THE
C      ROTATIONAL VELOCITIES IMPOSED ON THE
C      AIRCRAFT, IN THE ROTATIONAL OR AIRCRAFT COORDINATE
C      SYSTEM. (X ALONG THE NOSE, Y OUT THE RIGHT WING,
C      Z OUT THE BOTTOM OF THE AIRCRAFT.)
C *****
C      SUBROUTINE ACRVEL
C      COMMON /AEE3/ PSIDOT,THEDOT,ZHDOT
C      COMMON /AEE4/ PSINW,THENW,PHINW,PSIOLD,THEOLD,PHIOLD
C      COMMON /AEE5/ FWCACC,AN,MCONT
C      COMMON /AEE3/ UAC,VAC,AC,UDOTAC,VDOTAC,WDOTAC,P,Q,R
C      ANG1=THENW
C      ANG2=PHINW
C      P=PHIOLD-SIN(ANG1)*PSIDOT
C      Q=COS(ANG2)*THEDOT+COS(ANG1)*SIN(ANG2)*PSIDOT
C      R=-SIN(ANG2)*THEDOT+COS(ANG1)*COS(ANG2)*PSIDOT
C      RETURN
C      END

C *****
C SUBROUTINE ACACC
C      (AIRCRAFT ACCELERATIONS) PUTS TOGETHER THE LINEAR
C      ACCELERATIONS RESULTING FROM TRANSLATION THROUGH
C      SPACE AND THE ROTATIONAL ACCELERATIONS RESULTING
C      FROM MANEUVERING. WHAT APPEARS BELOW IS THE RHS
C      OF THE CLASSIC EQUATIONS OF MOTION FOR AN AIRCRAFT.
C *****
C      SUBROUTINE ACACC
C      COMMON /AEE2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
C      1ZTEMP,ZOLD
C      COMMON /AEE3/ AX,AY,AZ
C      COMMON /AEE5/ FWCACC,AN,MCONT
C      COMMON /AEE3/ UAC,VAC,AC,UDOTAC,VDOTAC,WDOTAC,P,Q,R
C      AX=UDOTAC-VAC*R+AC*Q
C      AY=VDOTAC+UAC*R+AC*P
C      AZ=WDOTAC-UAC*Q+VAC*P
C      A=-VAC*R
C      B=AC*Q
C      C=UAC*R
C      D=-AC*P
C      E=-UAC*Q
C      F=VAC*P
C      RETURN
C      END

C *****
C SUBROUTINE ACLMIS
C      ACLMIS (AIRCRAFT LIMITATIONS) COMPARES THE
C      ACCELERATIONS RESULTING FROM A PROPOSED MOVE ALONG A
C      FLIGHT-PATH TRAJECTORY TO THE PHYSICAL LIMITATIONS OF
C      THE AIRCRAFT. THIS IS ESSENTIALLY THE RHS OF THE
C      EQUATIONS OF MOTION. IF THE TRAJECTORY ACCELERATIONS
C      ARE LARGER THAN ALLOWABLE THEY ARE ADJUSTED DOWNWARD
C      IN MAGNITUDE UNTIL THE RHS AND LHS OF THE EQUATIONS
C      OF MOTION ARE EQUATED.
C *****
C      SUBROUTINE ACLMIS
C      COMMON /AEE3/ AX,AY,AZ
C      COMMON /AEE4/ THMAX,ANMAX,CLMAX,GEE,TO,RHOSSL,WL,CDO,CDK,AREA,DTDH
C      COMMON /AEE5/ FWCACC,AN,MCONT
C      CALL THRUST
C      IF (AN.LT.ANMAX.AND.FWCACC.LT.THMAX) CALL BOOSTB
C      IF (AN.GT.ANMAX.OR.FWCACC.GT.THMAX) CALL REDUCB
C      RETURN
C      END

```



```

C*****
C SUBROUTINE BCCSTR      ECCSTR INCREASES THE LINEAR ACCELERATIONS UNTIL
C                        MAXIMUM ALLOWABLE PERFORMANCE IS REACHED
C*****
      SUBROUTINE BCCSTR
      COMMON /AEC2/ XNEW,DELX,XTEMP,XCLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1 ZTEMP,ZOLD
      COMMON /AEC4/ AG1,AH1,AI1,AJ1,AJ2
      COMMON /AEE1/ UDOT,VDOT,WDOT
      COMMON /AEE3/ AX,AY,AZ
      COMMON /AEE4/ THMAX,ANMAX,CLMAX,GEE,TO,BHOSSL,WL,CDO,CDK,AREA,DTDH
      COMMON /AEE5/ FWDACC,AN,MCONT
      MCONT=1
10  CCNTINUE
      I=0
      IF (AG1.EQ.DELX.OR.AG1.EQ.0.) GO TO 20
      IF (ABS(AG1).LT.AES(AH1).AND.AES(AG1).LT.ABS(AI1)) I=1
      IF (I.NE.1) GO TO 20
      UDOT=1.02*UDOT
      CONTINUE
20  IF (AH1.EQ.DELY.OR.AH1.EQ.0.) GO TO 30
      IF (ABS(AH1).LT.AES(AH1).AND.AES(AH1).LT.ABS(AI1)) I=2
      IF (I.NE.2) GO TO 30
      VDOT=1.02*VDOT
      CONTINUE
30  IF (AI1.EQ.DELZ.OR.AI1.EQ.0.) GO TO 40
      IF (ABS(AI1).LT.AES(AH1).AND.AES(AI1).LT.ABS(AH1)) I=3
      IF (I.NE.3) GO TO 40
      WDOT=1.02*WDOT
      CONTINUE
40  IF (AG1.EQ.DELX.OR.AG1.EQ.0.) GO TO 50
      IF (ABS(AG1).EQ.AES(AI1).AND.AES(AG1).LT.ABS(AH1)) I=4
      IF (I.NE.4) GO TO 50
      UDOT=1.02*UDOT
      WDOT=1.02*WDOT
      CONTINUE
50  IF (AH1.EQ.DELY.OR.AH1.EQ.0.) GO TO 60
      IF (ABS(AH1).EQ.AES(AH1).AND.AES(AG1).LT.ABS(AI1)) I=5
      IF (I.NE.5) GO TO 60
      UDOT=1.02*UDOT
      VDOT=1.02*VDOT
      CONTINUE
60  IF (AI1.EQ.DELZ.OR.AI1.EQ.0.) GO TO 70
      IF (ABS(AH1).EQ.AES(AI1).AND.AES(AH1).LT.ABS(AG1)) I=6
      IF (I.NE.6) GO TO 70
      VDOT=1.02*VDOT
      WDOT=1.02*WDOT
      CONTINUE
70  CALL SMLCAL
      CALL THRUST
      IF (AN.GE.ANMAX.OR.FWDACC.GE.THMAX) GO TO 80
      GO TO 10
      CONTINUE
80  IF (AN.GT.ANMAX) AN=ANMAX
      IF (FWDACC.GT.THMAX) FWDACC=THMAX
      RETURN
      END

```



```

C*****
C SUBROUTINE REDUCR
C      REDUCR IS CALL FROM ACLMTS AND REDUCES THE "NEW"
C      VELOCITY AND ULTIMATELY POSITION TO SET LOADS
C      ON THE AIRCRAFT WITHIN LIMITS.
C*****
      SUBROUTINE REDUCR
      COMMON /AEC3/ VELCN, VELOT, VELCC, N, UNEN, UTEMP, UOLD, VNEW, VTEMP, VOLD,
1 WNEW, WTEMP, WOLD
      COMMON /AEC4/ AG1, AH1, AI1, AJ1, AJ2
      COMMON /AED2/ VSIAL1, VMAX, VMAX1, VMAX2, DELTEE, DELTV, DELTZ
      COMMON /AEE3/ AX, AY, AZ
      COMMON /ABE4/ THMAX, ANMAX, CLMAX, GEE, TO, SHOSSL, AL, CDO, CDK, AREA, DTCH
      COMMON /AEE5/ FWDACC, AN, MCONT
      COMMON /LCC/ X1, Y1, Z1, V1, ITR, ITR2
      MCONT=1
10  CONTINUE
      VELON=VELCN-DELTV
      A=VSTALL+2.*DELTV
      IF (VELON.GE.A) GC TO 20
      ANG=ACOS(AG1/AJ2)
      AJ2=1.1*AJ2
      AG1=COS(ANG)*AJ2
      AH1=SIN(ANG)*AJ2
      VELON=V1
20  CCONTINUE
      N=0
      CALL MVELCS
      CALL MVDCTS
      CALL SMLCAL
      CALL THRUST
      IF (AN.LE.ANMAX.AND.FWDACC.LE.IHMAX) GO TO 30
      GO TO 10
30  CONTINUE
      RETURN
      END

```

```

C*****
C SUBROUTINE SMLCAL
C      (SMALL CALL) IS A SMALLER VERSION OF (BIG CALL) AND
C      USED IN THE ITERATIVE LOOPS OF (AIRCRAFT LIMITATIONS)
C*****
      SUBROUTINE SMLCAL
      CALL DELVEC
      CALL TMPENT
      CALL TMPVEL
      CALL INRTM
      CALL ANGLES
      CALL ROTRAT
      CALL ACTERM1
      CALL ACTERM2
      CALL ACTERM3
      CALL ACBVEL
      CALL ACACC
      RETURN
      END

```



```

C *****
C SUBROUTINE THRUST
C THRUST IS USED BY (AIRCRAFT LIMITATIONS) TO COMPUTE
C THE AVAILABLE FORWARD ACCELERATION BASED UPON THRUST
C DRAG, G-LOADING, POSITION IN SPACE AND GRAVITY.
C *****

```

```

SUBROUTINE THRUST
COMMON /ABC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1WNEW,WTEMP,WOLD
COMMON /ABD4/ PSINew,THENew,PHINew,PSIOld,THEOld,PHIOld
COMMON /ABE3/ AX,AY,AZ
COMMON /ABE4/ THMAX,ANMAX,CLMAX,GEE,TC,PHOSSL,AL,CDO,CDK,AREA,DTEH
COMMON /ABE5/ FWLACC,AN,MCONT
ANG=THENew
THRQU=AX/GEE
AN=SQRT(AY**2+AZ**2)/GEE
SIGMA=(1+CTDH*ZTEMP/TO)**4.257
RHOALT=RHCSSL*SIGMA
CL=(2.*AN*WL)/(RHOALT*VELCT**2)
DRAG=AN*((CDO+CDK*CL**2)/CL)
FWLACC=SIN(ANG)+THRQU+DRAG
RETURN
END

```

```

C *****
C SUBROUTINE XCHANGE
C ONCE THE EQUATIONS OF MOTION ARE SATISFIED AT THE
C PROPOSED LOCATION (TEMP) THE TEMP LOCATION BECOMES
C THE CURRENT LOCATION (OLD).
C *****

```

```

SUBROUTINE XCHANGE
COMMON /ABC2/ XNEW,DELX,XTEMP,XOLD,YNEW,DELY,YTEMP,YOLD,ZNEW,DELZ,
1ZTEMP,ZOLD
COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,UOLD,VNEW,VTEMP,VOLD,
1WNEW,WTEMP,WOLD
COMMON /ABD4/ PSINew,THENew,PHINew,PSIOld,THEOld,PHIOld
COMMON /ABE5/ AD1,AE1,AF1,AF2
XOLD=XTEMP
YOLD=YTEMP
ZOLD=ZTEMP
UOLD=UTEMP
VOLD=VTEMP
WOLD=WTEMP
PSIOld=PSINew
THEOld=THENew
PHIOld=PHINew
VELOO=VELOT
RETURN
END

```



```

C*****
C SUBROUTINE WAYPNT THE RELEVANT PARAMETERS FOR THE CURRENT POINT ARE
C ASSIGNED.
C*****
SUBROUTINE WAYPNT
COMMON /AEC1/ WX(1000), WY(1000), WZ(1000), WU(1000), WV(1000), WW(1000)
1) , WH(1000), WP(1000), WR(1000), WG(1000), T(1000), ICT
COMMON /AEC2/ XNEW, DELX, XTEMP, XOLD, YNEW, DELY, YTEMP, YOLD, ZNEW, DELZ,
12TEMP, ZOLD
COMMON /AEC3/ VELCN, VELOT, VELCC, N, UNEW, UTEMP, UOLD, VNEW, VTEMP, VOLD,
13NEW, VTEMP, ZOLD
COMMON /AEC4/ VSTALL, VMAX, VMAX1, VMAX2, DELTEE, DELTY, DELTAZ
COMMON /AEC5/ PSINew, THENew, PHINew, PSIOLD, THEOLD, PHIOLD
COMMON /AEC6/ FWDACC, AN, ACONT
WX(ICT) = XOLD
WY(ICT) = YOLD
WZ(ICT) = ZOLD
WU(ICT) = UOLD
WV(ICT) = VOLD
WW(ICT) = WOLD
WH(ICT) = -PSINew
WP(ICT) = THENew
WR(ICT) = -PHINew
WG(ICT) = AN
T(ICT) = T(ICT-1) + DELTEE
RETURN
END

C*****
C SUBROUTINE FILE THE CURRENT POINT IS IMMEDIATELY SAVED.
C*****
SUBROUTINE FILE
COMMON /AEC1/ WX(1000), WY(1000), WZ(1000), WU(1000), WV(1000), WW(1000)
1) , WH(1000), WP(1000), WR(1000), WG(1000), T(1000), ICT
COMMON /AEC2/ XNEW, DELX, XTEMP, XOLD, YNEW, DELY, YTEMP, YOLD, ZNEW, DELZ,
12TEMP, ZOLD
COMMON /LCC/ X1, Y1, Z1, V1, ITB, LTR2
IF (XOLD.NE.XNEW.OR.YOLD.NE.YNEW.OR.ZOLD.NE.ZNEW) GO TO 10
WRITE (11,20) T(ICT), WX(ICT), WY(ICT), WZ(ICT), WU(ICT), WV(ICT), WW(IC
1T), WH(ICT), WP(ICT), WR(ICT), WG(ICT), LTR
RETURN
CONTINUE
LTR=0
WRITE (11,20) T(ICT), WX(ICT), WY(ICT), WZ(ICT), WU(ICT), WV(ICT), WW(IC
1T), WH(ICT), WP(ICT), WR(ICT), WG(ICT), LTR
RETURN
20 FORMAT (11(E10.4, 1X), I3)
END

```



```

C *****
C BLOCK DATA
C *** CAUTION ** THIS BLOCK DATA IS AN ABBREVIATED
C VERSION. -- D.A.M. -- 16MAR84
C *****
C BLOCK DATA
COMMON /ICC/ X1,Y1,Z1,V1,IT5,ITR2
COMMON /MN1/ MINX,MAXX,MINY,MAXY,MINX1,MAXX1,MINX2,MAXX2
COMMON /MN/ IBAUD
COMMON /PAR2/ XGUN(7),YGUN(7),ZGUN(7),XSAM,YSAM,ZSAM,GR(7)
COMMON /PAR4/ APFMAY,HTMIN,HTMAX,POPMIN,BAALT,SRMIN,BRMAX,BRPMIN,B
1BRMAX,ITL,AMTF
COMMON /ABC1/ WX(1000),WY(1000),WZ(1000),WU(1000),WV(1000),WW(1000
1),WH(1000),WP(1000),WR(1000),WG(1000),I(1000),ICT
COMMON /ABC2/ XNEW,DELT,XTEMP,XCLD,YNEW,DELT,YTEMP,YOLD,ZNEW,DELT,
1ZTEMP,ZOLD
COMMON /ABC3/ VELCN,VELOT,VELCC,N,UNEW,UTEMP,JOLD,VNEW,VTEMP,VOLD,
1VNEW,VTEMP,VOLD
COMMON /ABD1/ AA1,AB1,AC1
COMMON /ABD2/ VSTALI,VMAX,VMAX1,VMAX2,DELTEE,DELTV,DELTAZ
COMMON /ABD3/ PSILOT,THEDOT,PHIDOT
COMMON /ABD4/ PSINEW,THENEW,PHINEW,PSIOLD,THEOLD,PHIOLD
COMMON /ABD5/ AD1,AE1,AF1,AF2
COMMON /ABE4/ THMAX,ANMAX,CLMAX,GEE,TC,PHOSSL,AL,CDO,CDK,AREA,OTEN
COMMON /TAB/ TARGX,TARGY
C-----
C FLASH DRIVER
C-----
C DATA IBAUD/1/
C-----
C MAE WINDOW X COORDINATES
C-----
C DATA MINX/100/
C DATA MAXX/3700/
C-----
C ALTIMETER WINDOW X COORDINATES
C-----
C DATA MINX1/3725/
C DATA MAXX1/3925/
C-----
C FREIGHTING WINDOW X COORDINATES
C-----
C DATA MINX2/3950/
C DATA MAXX2/4000/
C-----
C COMMON Y COORDINATES FOR THE THREE WINDOWS
C-----
C DATA MINY/400/
C DATA MAXY/2800/
C-----
C PREDETERMINED WEAPON EMPLACEMENT PARAMETERS
C-----
C DATA XGUN/14800.,16200.,13600.,13400.,11300.,15600.,12300./
C DATA YGUN/9600.,3200.,7200.,8000.,2700.,10900.,7500./
C DATA ZGUN/40.,40.,20.,20.,50.,90.,20./
C DATA XSAM/12000./,YSAM/6000./,ZSAM/50./
C DATA GR/1000.,1000.,1400.,1400.,2500.,2500.,2500./
C-----
C TARGET COORDINATES
C-----
C DATA TARGX/14000./,TARGY/7220./
C-----
C CONVERSION FACTORS: RADIANS TO DEGREES
C METERS TO FEET
C-----
C DATA RTD/57.29578/
C DATA AMTF/3.28084/
C-----
C GRAVITATIONAL CONSTANT (M/S2)
C-----
C DATA GEE/9.807/
C-----
C SIMULATION PARAMETERS
C-----
C-----
C CARDINAL COMPASS HEADINGS (RADIANS)

```



```

C-----
C DATA AA1/1.570796//,AB1/3.141593//,AC1/4.7124/
C-----
C PITCH ROLL AND YAW PARAMETERS
C-----
C DATA AD1/.5236//,AE1/.0524//,AF1/.6503//,AF2/2.0944/
C-----
C MAXIMUM THRUST OF THE AIRCRAFT (DIMENSIONLESS)
C-----
C DATA THMAX/0.40/
C-----
C MAXIMUM AIRCRAFT GEE-LOADING (DIMENSIONLESS)
C-----
C DATA ANMAX/6./
C-----
C RHOSSL IS THE DENSITY AT STANDARD SEA-LEVEL CONDITIONS. IT HAS
C DIMENSIONS OF (N.S2/M4)
C-----
C DATA RHOSSL/1.225/
C-----
C STANDARD SEA-LEVEL TEMPERATURE (DEGREES K)
C-----
C DATA T0/288.16/
C-----
C TEMPERATURE LAPSE RATE FOR THE TROPOSPHERE (DEG K/M)
C-----
C DATA DTDR/-0.5E-3/
C-----
C MAXIMUM WING LOADING (N/M2)
C-----
C DATA WL/4788./
C-----
C SPEED BRAKE AREA: DEFINED AS A PERCENTAGE OF TOTAL WING AREA
C-----
C DATA AREA/.05/
C-----
C MAXIMUM COEFFICIENT OF LIFT
C-----
C DATA CLMAX/1./
C-----
C COEFFICIENT OF DRAG AT ZERO ANGLE OF ATTACK
C-----
C DATA CDO/.015/
C-----
C PROFILE DRAG COEFFICIENT
C-----
C DATA CDK/0.1/
C-----
C MINIMUM ALTITUDE FOR THE GAME (M)
C-----
C DATA HTMIN/60./
C-----
C MAXIMUM ALTITUDE APPROACHING THE TARGET (M)
C-----
C DATA APPMAX/457./
C-----
C MAXIMUM ALTITUDE AFTER ENTERING POP-UP RANGE OF 6 KM. (M)
C-----
C DATA HTMAX/2050./
C-----
C MAXIMUM VELOCITIES BEFORE AND AFTER BOMB RELEASE (M/S)
C-----
C DATA VMAX1/260./,VMAX2/310./
C-----
C STALL VELOCITY (M/S)
C-----
C DATA VSTALL/90./
C-----
C MAXIMUM RANGE (METERS) A POP-UP MAY BE COMMENCED
C-----
C DATA PCPMIN/6000./
C-----
C THE FOLLOWING ARE TEMPORARY PARAMETERS FOR THE INITIAL CONDITIONS OF
C THE AIRCRAFT. LATER THIS WILL BE ENTERED INTERACTIVELY.
C-----
C DATA TX/0./,TY/6000./,HZ/200./,WU/200./,WV/0./,WW/0./

```



```

C-----DATA WH/1.5708//,WP/0.//,WR/0.//,WG/0.//,T/0.//
C INCREMENT OF VELOCITY CHANGE (5 KNCT INC CONVERTED TO M/S)
C-----
C-----DATA DELTV/3.//
C INCREMENT OF TIME CHANGE(SECONDS)
C-----
C-----DATA DELTEE/1.00//
C INCREMENT OF ALTITUDE CHANGE(50 FT INC CONVERTED TO METERS)
C-----
C-----DATA DELTAZ/15.24//
C-----
C TEMECRABY PROGRAM INITIALIZATION
C last data line simulates input from the graphics buffer through
C the ibmpip subroutines xypt and zvpt.
C-----
DATA XOLD/0.//,YOLD/6000.//,ZOLD/200.//
DATA UOLD/200.//,VCID/0.//,WOLD/0.//
DATA PSICLD/0.//,TEEOLD/0.//,PHICLD/0.//
DATA VELCC/200.//
DATA X1/6000.//,Y1/8000.//,Z1/300.//,V1/260.//
END

```


LIST OF REFERENCES

1. Johns, E.B., Creation of a Transportable Interactive User Interface for Improved AAA Simulation Computer Program, Master's Thesis, Naval Postgraduate School, Monterey, California, 1982.
2. International Business Machines Corporation, IBM 3277 Graphics Attachment Support (Programming RPO P09013): Program Reference and Operations Manual SA20-2137-1, pp. 206-213, IBM.
3. Meriam, J.L., Dynamics, 2nd ed., pp.17-20, Wiley, 1971.
4. Hurt, H.H. Jr., Aerodynamics for Naval Aviators, pp.176-178, U.S. Government Printing Office, 1980.
5. Bell, R.W., Performance and Static Stability (AE2036), class notes for course at Naval Postgraduate School, Monterey, California, 12 July 1982.
6. Roskam, J., Airplane Flight Dynamics and Automatic Flight Controls, part 1, pp. 9-33, Roskam Aviation and Engineering Corporation, 1979.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria Va 22314	2
2. Library, Code 0142 Naval Postgraduate School Monterey Ca 93943	2
3. Professor R.E. Ball Code 67Bp Department of Aeronautics Naval Postgraduate School Monterey, Ca 93943	2
4. LT D.A. Milton FAWFRA Box 58 FPO San Francisco, Ca 96654	1
5. Combat Data Information Center AFWAL/FIES/CDIC Wright-Patterson AFB, Ohio 45433	1
6. Mr. Martin Lentz ASD/ENFTV Wright-Patterson AFB, Ohio 45433	1

207454

Thesis
M59726 Milton
c.1 Improvements to
IBMPIP.

207454

Thesis
M59726 Milton
c.1 Improvements to
IBMPIP.

thesM59726

Improvements to IBMPIP.



3 2768 001 89088 2

DUDLEY KNOX LIBRARY